Серия учебных пособий

Информатика в техническом университете

И.П. Норенков

АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ

Москва — 2000

Даны сведения по различным аспектам и видам обеспечения систем автоматизированного проектирования, необходимые квалифицированным пользователям САПР в различных областях техники. Значительное внимание уделено математическому обеспечению процедур анализа и синтеза проектных решений, построению локальных и корпоративных вычислительных сетей САПР, составу и функциям системных сред САПР. Освещены также активно развиваемые в последнее время методики концептуального проектирования сложных систем, положенные в основу технологии CALS, а также вопросы интеграции САПР с автоматизированными системами управления и делопроизводства.

Книга предназначена для студентов технических высших учебных заведений, может быть полезна аспирантам и работникам промышленности, использующим методы и средства САПР в своей работе.

The textbook is devoted to basic problems and aspects of CAE/CAD/CAM systems. Main approaches to design procedures including methods and algorithms for modeling, simulation and synthesis are considered. The information about workstations, networks, design frameworks and CALS components are given.

The book is intended to students of technological universities, it may be recommended to postgraduates and industry specialists using CAE/CAD/CAM in their activity.

Оглавление

Предисловие

Автоматизация проектирования занимает особое место среди информационных технологий.

Во-первых, автоматизация проектирования — синтетическая дисциплина, ее составными частями являются многие другие современные информационные технологии. Так, техническое обеспечение систем автоматизированного проектирования (САПР) основано на использовании вычислительных сетей и телекоммуникационных технологий, в САПР используются персональные компьютеры и рабочие станции, есть примеры применения мейнфреймов. Математическое обеспечение САПР отличается богатством и разнообразием используемых методов вычислительной математики, статистики, математического программирования, дискретной математики, искусственного интеллекта. Программные комплексы САПР относятся к числу наиболее сложных современных программных систем, основанных на операционных системах Unix, Windows-95/NT, языках программирования С, С++, Java и других, современных CASE-технологиях, реляционных и объектно-ориентированных системах управления базами данных (СУБД), стандартах открытых систем и обмена данными в компьютерных средах.

Во-вторых, знание основ автоматизации проектирования и умение работать со средствами САПР требуется практически любому инженеру-разработчику. Компьютерами насыщены проектные подразделения, конструкторские бюро и офисы. Работа конструктора за обычным кульманом, расчеты с помощью логарифмической линейки или оформление отчета на пишущей машинке стали анахронизмом. Предприятия, ведущие разработки без САПР или лишь с малой степенью их использования, оказываются неконкурентоспособными как из-за больших материальных и временных затрат на проектирование, так и из-за невысокого качества проектов.

Появление первых программ для автоматизации проектирования за рубежом и в СССР относится к началу 60-х гг. Тогда были созданы программы для решения задач строительной механики, анализа электронных схем, проектирования печатных плат. Дальнейшее развитие САПР шло по пути создания аппаратных и программных средств машинной графики, повышения вычислительной эффективности программ моделирования и анализа, расширения областей применения САПР, упрощения пользовательского интерфейса, внедрения в САПР элементов искусственного интеллекта.

К настоящему времени создано большое число программно-методических комплексов для САПР с различными степенью специализации и прикладной ориентацией. В результате автоматизация проектирования стала необходимой составной частью подготовки инженеров разных специальностей; инженер, не владеющий знаниями и не умеющий работать в САПР, не может считаться полноценным специалистом.

Подготовка инженеров разных специальностей в области САПР включает базовую и специальную компоненты. Наиболее общие положения, модели и методики автоматизированного проектирования входят в программу курса, посвященного основам САПР, более детальное изучение тех методов и программ, которые специфичны для конкретных специальностей, предусматривается в профильных дисциплинах.

Данный учебник ориентирован на базовую подготовку студентов различных инженерных специальностей в области САПР.

Глава 1 является вводной. Здесь даны начальные сведения о процессе проектирования технических объектов, изложены основные понятия системотехники, пояснены структура САПР и ее место в ряду других промышленных автоматизированных систем.

Глава 2 посвящена техническому обеспечению САПР, основное внимание уделено локальным и корпоративным вычислительным сетям. Рассмотрены наиболее распространенные типы локальных сетей, методы доступа, протоколы и характеристики каналов передачи данных в вычислительных сетях.

В **главе 3** содержатся сведения о моделях и методах, используемых для анализа проектных решений на различных иерархических уровнях, начиная с метода конечных элементов для анализа полей физических величин и кончая основами имитационного моделирования систем массового обслуживания. Кратко изложены подходы к геометрическому моделированию и обработке графической информации для ее визуализации.

Методы параметрического и структурного синтеза проектных решений изложены в **главе 4**. Дан обзор критериев оптимальности и методов математического программирования для расчета оптимальных значений проектных параметров. Пояснены трудности формализации структурного синтеза и охарактеризованы перспективные методы его выполнения.

В **главе 5** представлена общая структура программного и информационного обеспечения САПР. Основное внимание уделено обслуживающим подсистемам, в том числе CASE-подсистемам разработки программного обеспечения.

Глава 6 знакомит читателя с современными средствами концептуального проектирования сложных систем, с подходами к созданию интегрированных систем проектирования и управления на базе IDEF методик и стандартов STEP.

В **приложении** приведены краткие сведения о важных международных стандартах в области информационной поддержки проектирования и производства промышленной продукции.



Введение в автоматизированное проектирование

1.1. Системный подход к проектированию

Понятие инженерного проектирования. *Проектирование* технического объекта — создание, преобразование и представление в принятой форме образа этого еще не существующего объекта. Образ объекта или его составных частей может создаваться в воображении человека в результате творческого процесса или генерироваться в соответствии с некоторыми алгоритмами в процессе взаимодействия человека и ЭВМ. В любом случае инженерное проектирование начинается при наличии выраженной потребности общества в некоторых технических объектах, которыми могут быть объекты строительства, промышленные изделия или процессы. Проектирование включает в себя разработку технического предложения и (или) технического задания (ТЗ), отражающих эти потребности, и реализацию ТЗ в виде проектной документации.

Обычно ТЗ представляют в виде некоторых документов, и оно является *исходным (первичным) описанием объекта*. Результатом проектирования, как правило, служит полный комплект документации, содержащий достаточные сведения для изготовления объекта в заданных условиях. Эта документация и есть *проект*, точнее *окончательное описание* объекта. Более коротко, проектирование — процесс, заключающийся в получении и преобразовании исходного описания объекта в окончательное описание на основе выполнения комплекса работ исследовательского, расчетного и конструкторского характера.

Преобразование исходного описания в окончательное порождает ряд промежуточных описаний, подводящих итоги решения некоторых задач и используемых для обсуждения и принятия проектных решений для окончания или продолжения проектирования.

Проектирование, при котором все проектные решения или их часть получают путем взаимодействия человека и ЭВМ, называют *автоматизированным*, в отличие от *ручного* (без использования ЭВМ) или *автоматического* (без участия человека на промежуточных этапах). Система, реализующая автоматизированное проектирование, представляет собой *систему автоматизированного проектирования* (в англоязычном написании *CAD System — Computer Aided Design System*).

Автоматическое проектирование возможно лишь в отдельных частных случаях для сравнительно несложных объектов. Превалирующим в настоящее время является автоматизированное проектирование.

Проектирование сложных объектов основано на применении идей и принципов, изложенных в ряде теорий и подходов. Наиболее общим подходом является системный подход, идеями которого пронизаны различные методики проектирования сложных систем.

Принципы системного подхода. Основные идеи и принципы проектирования сложных систем выражены в системном подходе. Для специалиста в области системотехники они являются очевидными и естественными, однако их соблюдение и реализация зачастую сопряжены с определенными трудностями, обусловливаемыми особенностями проектирования. Как и большинство взрослых образованных людей, правильно использующих родной язык без привлечения правил грамматики, инженеры используют системный подход без обращения к пособиям по системному анализу. Однако интуитивный подход без применения правил системного анализа может оказаться недостаточным для решения все более усложняющихся задач инженерной деятельности.

Основной общий принцип системного подхода заключается в рассмотрении частей явления или сложной системы с учетом их взаимодействия. *Системный подход включает в себя выявление структуры системы, типизацию связей, определение атрибутов, анализ влияния внешней среды.*

Системный подход рассматривают как направление научного познания и социальной политики. Он является базой для обобщающей дисциплины “*Теория систем*” (другое используемое название — “*Системный анализ*”). *Теория систем — дисциплина, в которой конкретизируются положения системного подхода; она посвящена исследованию и проектированию сложных экономических, социальных, технических систем, чаще всего слабоструктурированных.* Характерными примерами таких систем являются производственные системы. При проектировании систем цели достигаются в многошаговых процессах принятия решений. Методы принятия решений часто выделяют в самостоятельную дисциплину, называемую “*Теория принятия решений*”.

В технике дисциплину, в которой исследуются сложные технические системы, их проектирование, и аналогичную теории систем, чаще называют *системотехникой*. *Предметом системотехники являются, во-первых, организация процесса создания, использования и развития технических систем, во-вторых, методы и принципы их проектирования и исследования.* В системотехнике важно уметь сформулировать цели системы и организовать ее рассмотрение с позиций поставленных целей. Тогда можно отбросить лишние и малозначимые части при проектировании и моделировании, перейти к постановке оптимизационных задач.

Системы автоматизированного проектирования и управления относятся к числу наиболее сложных современных искусственных систем. Их проектирование и сопровождение невозможны без системного подхода. Поэтому идеи и положения системотехники входят составной частью в дисциплины, посвященные изучению современных автоматизированных систем и технологий их применения. Интерпретация и конкретизация системного подхода имеют место в ряде известных подходов с другими названиями, которые также можно рассматривать как компоненты системотехники. Таковы структурный, блочно-иерархический, объектно-ориентированный подходы.

При *структурном подходе*, как разновидности системного, требуется синтезировать варианты системы из компонентов (блоков) и оценивать варианты при их частичном переборе с предварительным прогнозированием характеристик компонентов.

Блочно-иерархический подход к проектированию использует идеи декомпозиции сложных описаний объектов и соответственно средств их создания на иерархические уровни и аспекты, вводит понятие стиля проектирования (восходящее и нисходящее), устанавливает связь между параметрами соседних иерархических уровней.

Ряд важных структурных принципов, используемых при разработке информационных систем и прежде всего их программного обеспечения (ПО), выражен в *объектно-ориентированном подходе* к проектированию (ООП). Такой подход имеет следующие преимущества в решении проблем управления сложностью и интеграции ПО: 1) вносит в модели приложений большую структурную определенность, распределяя представленные в приложении данные и процедуры между классами объектов; 2) сокращает объем спецификаций, благодаря введению в описания иерархии объектов и отношений наследования между свойствами объектов разных уровней иерархии; 3) уменьшает вероятность искажения данных вследствие ошибочных действий за счет ограничения доступа к определенным категориям данных в объектах. Описание в каждом классе объектов допустимых обращений к ним и принятых форматов сообщений облегчает согласование и интеграцию ПО.

Для всех подходов к проектированию сложных систем характерны также следующие особенности.

1. *Структуризация* процесса проектирования, выражаемая декомпозицией проектных задач и документации, выделением стадий, этапов, проектных процедур. Эта структуризация является сущностью блочно-иерархического подхода к проектированию.

2. *Итерационный* характер проектирования.

3. *Типизация* и *унификация* проектных решений и средств проектирования.

Основные понятия системотехники. В теории систем и системотехнике введен ряд терминов, среди них к базовым нужно отнести следующие понятия.

Система — множество элементов, находящихся в отношениях и связях между собой.

Элемент — такая часть системы, представление о которой нецелесообразно подвергать при проектировании дальнейшему членению.

Сложная система — система, характеризуемая большим числом элементов и, что наиболее важно, большим числом взаимосвязей элементов. Сложность системы определяется также видом взаимосвязей элементов, свойствами *целенаправленности, целостности, членимости, иерархичности, многоаспектности*. Очевидно, что современные автоматизированные информационные системы и, в частности, системы автоматизированного проектирования, являются сложными в силу наличия у них перечисленных свойств и признаков.

Подсистема — часть системы (подмножество элементов и их взаимосвязей), которая имеет свойства системы.

Надсистема — система, по отношению к которой рассматриваемая система является подсистемой.

Структура — отображение совокупности элементов системы и их взаимосвязей; понятие структуры отличается от понятия самой системы также тем, что при описании структуры принимают во внимание лишь типы элементов и связей без конкретизации значений их параметров.

Параметр — величина, выражающая свойство или системы, или ее части, или влияющей на систему среды. Обычно в моделях систем в качестве параметров рассматривают величины, не изменяющиеся в процессе исследования системы. Параметры подразделяют на *внешние, внутренние* и *выходные*, выражающие свойства элементов системы, самой системы, внешней среды соответственно. Векторы внутренних, выходных и внешних параметров далее обозначаются $\mathbf{X} = (x_1,x_2...x_n)$, $\mathbf{Y} = (y_1,y_2...y_m)$, $\mathbf{Q} = (q_1,q_2,...q_k)$ соответственно.

Фазовая переменная — величина, характеризующая энергетическое или информационное наполнение элемента или подсистемы.

Состояние — совокупность значений фазовых переменных, зафиксированных в одной временной точке процесса функционирования.

Поведение (динамика) системы — изменение состояния системы в процессе функционирования.

Система без последействия — ее поведение при $t > t_0$ определяется заданием состояния в момент t_0 и вектором внешних воздействий $\mathbf{Q}(t)$. В системах с последействием, кроме того, нужно знать предысторию поведения, т.е. состояния системы в моменты, предшествующие t_0 .

Вектор переменных V*, характеризующих состояние* (вектор переменных состояния), — неизбыточное множество фазовых переменных, задание значений которых в некоторый момент времени полностью определяет поведение системы в дальнейшем (в автономных системах без последействия).

Пространство состояний — множество возможных значений вектора переменных состояния.

Фазовая траектория — представление процесса (зависимости $\mathbf{V}(t)$) в виде последовательности точек в пространстве состояний.

К характеристикам сложных систем, как сказано выше, часто относят следующие понятия.

Целенаправленность — свойство искусственной системы, выражающее назначение системы. Это свойство необходимо для оценки эффективности вариантов системы.

Целостность — свойство системы, характеризующее взаимосвязанность элементов и наличие зависимости выходных параметров от параметров элементов, при этом большинство выходных параметров не является простым повторением или суммой параметров элементов.

Иерархичность — свойство сложной системы, выражающее возможность и целесообразность ее иерархического описания, т.е. представления в виде нескольких уровней, между компонентами которых имеются отношения целое-часть.

Составными частями системотехники являются следующие основные разделы:

— иерархическая структура систем, организация их проектирования;

— анализ и моделирование систем;

— синтез и оптимизация систем.

Моделирование имеет две четко различимые задачи: 1 — создание моделей сложных систем (в англоязычном написании — *modeling*); 2 — анализ свойств систем на основе исследования их моделей (*simulation*).

Синтез также подразделяют на две задачи: 1 — синтез структуры проектируемых систем (*структурный синтез*); 2 — выбор численных значений параметров элементов систем (*параметрический синтез*). Эти задачи относятся к области принятия проектных решений.

Моделирование и оптимизацию желательно выполнять с учетом статистической природы систем. Детерминированность — лишь частный случай. При проектировании характерны нехватка достоверных исходных данных, неопределенность условий принятия решений. Учет статистического характера данных при моделировании в значительной мере основан на методе статистических испытаний (методе Монте-Карло), а принятие решений — на использовании нечетких множеств, экспертных систем, эволюционных вычислений.

Пример 1. Компьютер является сложной системой в силу наличия у него большого числа элементов, разнообразных связей между элементами и подсистемами, свойств целенаправленности, целостности, иерархичности. К подсистемам компьютера относятся процессор (процессоры), оперативная память, кэш-память, шины, устройства ввода-вывода.

В качестве надсистемы могут выступать вычислительная сеть, автоматизированная и (или) организационная система, к которым принадлежит компьютер. Внутренние параметры — времена выполнения арифметических операций, чтения (записи) в накопителях, пропускная способность шин и др. Выходные параметры — производительность компьютера, емкость оперативной и внешней памяти, себестоимость, время наработки на отказ и др. Внешние параметры — напряжение питания сети и его стабильность, температура окружающей среды и др.

Пример 2. Для двигателя внутреннего сгорания подсистемами являются коленчатый вал, механизм газораспределения, поршневая группа, система смазки и охлаждения. Внутренние параметры — число цилиндров, объем камеры сгорания и др. Выходные параметры — мощность двигателя, КПД, расход топлива и др. Внешние параметры — характеристики топлива, температура воздуха, нагрузка на выходном валу.

Пример 3. Подсистемы электронного усилителя — усилительные каскады; внутренние параметры — сопротивления резисторов, емкости конденсаторов, параметры транзисторов; выходные параметры — коэффициент усиления на средних частотах, полоса пропускания, входное сопротивление; внешние параметры — температура окружающей среды, напряжения источников питания, сопротивление нагрузки.

1.2. Структура процесса проектирования

Иерархическая структура проектных спецификаций и иерархические уровни проектирования. При использовании блочно-иерархического подхода к проектированию представления о проектируемой системе расчленяют на *иерархические уровни*. На верхнем уровне используют наименее детализированное представление, отражающее только самые общие черты и особенности проектируемой системы. На следующих уровнях степень подробности описания возрастает, при этом рассматривают уже отдельные блоки системы, но с учетом воздействий на каждый из них его соседей. Такой подход позволяет на каждом иерархическом уровне формулировать задачи приемлемой сложности, поддающиеся решению с помощью имеющихся средств проектирования. Разбиение на уровни должно быть таким, чтобы документация на блок любого уровня была обозрима и воспринимаема одним человеком.

Другими словами, блочно-иерархический подход есть декомпозиционный подход (его можно назвать также диакоптическим), который основан на разбиении сложной задачи большой размерности на последовательно и (или) параллельно решаемые группы задач малой размерности, что существенно сокращает требования к используемым вычислительным ресурсам или время решения задач.

Можно говорить не только об иерархических уровнях спецификаций, но и об *иерархических уровнях проектирования,* понимая под каждым из них совокупность спецификаций некоторого иерархического уровня совместно с постановками задач, методами получения описаний и решения возникающих проектных задач.

Список иерархических уровней в каждом приложении может быть специфичным, но для большинства приложений характерно следующее наиболее крупное выделение уровней:

— *системный* уровень, на котором решают наиболее общие задачи проектирования систем, машин и процессов; результаты проектирования представляют в виде структурных схем, генеральных планов, схем размещения оборудования, диаграмм потоков данных и т.п.;

— *макроуровень*, на котором проектируют отдельные устройства, узлы машин и приборов; результаты представляют в виде функциональных, принципиальных и кинематических схем, сборочных чертежей и т.п.;

— *микроуровень*, на котором проектируют отдельные детали и элементы машин и приборов.

В каждом приложении число выделяемых уровней и их наименования могут быть различными. Так, в радиоэлектронике микроуровень часто называют компонентным, макроуровень — схемотехническим. Между схемотехническим и системным уровнями вводят уровень, называемый функционально-логическим. В вычислительной технике системный уровень подразделяют на уровни проектирования ЭВМ (вычислительных систем) и вычислительных сетей. В машиностроении имеются уровни деталей, узлов, машин, комплексов.

В зависимости от последовательности решения задач иерархических уровней различают нисходящее, восходящее и смешанное проектирование (стили проектирования). Последовательность решения задач от нижних уровней к верхним характеризует *восходящее* проектирование, обратная последовательность приводит к *нисходящему* проектированию, в *смешанном* стиле имеются элементы как восходящего, так и нисходящего проектирования. В большинстве случаев для сложных систем пред-

почитают нисходящее проектирование. Отметим однако, что при наличии заранее спроектированных составных блоков (устройств) можно говорить о смешанном проектировании.

Неопределенность и нечеткость исходных данных при нисходящем проектировании (так как еще не спроектированы компоненты) или исходных требований при восходящем проектировании (поскольку ТЗ имеется на всю систему, а не на ее части) обусловливают необходимость прогнозирования недостающих данных с последующим их уточнением, т.е. последовательного приближения к окончательному решению (*итерационность* проектирования).

Наряду с декомпозицией описаний на иерархические уровни применяют разделение представлений о проектируемых объектах на аспекты.

Аспект описания (страта) — описание системы или ее части с некоторой оговоренной точки зрения, определяемой функциональными, физическими или иного типа отношениями между свойствами и элементами.

Различают аспекты функциональный, информационный, структурный и поведенческий (процессный). *Функциональное* описание относят к функциям системы и чаще всего представляют его функциональными схемами. *Информационное* описание включает в себя основные понятия предметной области (сущности), словесное пояснение или числовые значения характеристик (атрибутов) используемых объектов, а также описание связей между этими понятиями и характеристиками. Информационные модели можно представлять графически (графы, диаграммы сущность-отношение), в виде таблиц или списков. *Структурное* описание относится к морфологии системы, характеризует составные части системы и их межсоединения и может быть представлено структурными схемами, а также различного рода конструкторской документацией. *Поведенческое* описание характеризует процессы функционирования (алгоритмы) системы и (или) технологические процессы создания системы. Иногда аспекты описаний связывают с подсистемами, функционирование которых основано на различных физических процессах.

Отметим, что в общем случае выделение страт может быть неоднозначным. Так, помимо указанного подхода. очевидна целесообразность выделения таких аспектов, как *функциональное* (разработка принципов действия, структурных, функциональных, принципиальных схем), *конструкторское* (определение форм и пространственного расположения компонентов изделий), *алгоритмическое* (разработка алгоритмов и программного обеспечения) и *технологическое* (разработка технологических процессов) проектирование систем. Примерами страт в случае САПР могут служить также рассматриваемые далее виды обеспечения автоматизированного проектирования.

Стадии проектирования. *Стадии* проектирования — наиболее крупные части проектирования, как процесса, развивающегося во времени. В общем случае выделяют стадии научно-исследовательских работ (НИР), эскизного проекта или опытно-конструкторских работ (ОКР), технического, рабочего проектов, испытаний опытных образцов или опытных партий. Стадию НИР иногда называют предпроектными исследованиями или стадией технического предложения. Очевидно, что по мере перехода от стадии к стадии степень подробности и тщательность проработки проекта возрастают, и рабочий проект уже должен быть вполне достаточным для изготовления опытных или серийных образцов. Близким к определению стадии, но менее четко оговоренным понятием, является понятие этапа проектирования.

Стадии (этапы) проектирования подразделяют на составные части, называемые *проектными процедурами*. Примерами проектных процедур могут служить подготовка деталировочных чертежей, анализ кинематики, моделирование переходного процесса, оптимизация параметров и другие проектные задачи. В свою очередь, проектные процедуры можно расчленить на более мелкие компоненты, называемые *проектными операциями*, например, при анализе прочности детали сеточными методами операциями могут быть построение сетки, выбор или расчет внешних воздействий, собственно моделирование полей напряжений и деформаций, представление результатов моделирования в графической и текстовой формах. Проектирование сводится к выполнению некоторых последовательностей проектных процедур — *маршрутов проектирования.*

Иногда разработку ТЗ на проектирование называют *внешним* проектированием, а реализацию ТЗ — *внутренним* проектированием.

Содержание технических заданий на проектирование. В ТЗ на проектирование объекта указывают, по крайней мере, следующие данные.

1. Назначение объекта.

2. Условия эксплуатации. Наряду с качественными характеристиками (представленными в вербальной форме) имеются числовые параметры, называемые *внешними* параметрами, для которых указаны области допустимых значений. Примеры внешних параметров: температура окружающей среды, внешние силы, электрические напряжения, нагрузки и т.п.

3. Требования к *выходным* параметрам, т.е. к величинам, характеризующим свойства объекта, интересующие потребителя. Эти требования выражены в виде *условий работоспособности*

$$y_i \; R \; T_i,$$

где y_i — i-й выходной параметр, $R \in$ {равно, меньше, больше, больше или равно, меньше или равно} — вид отношения; T_i — норма i-го выходного параметра. В случае R = “равно” нужно задать требуемую точность выполнения равенства.

Примеры условий работоспособности:

расход топлива на 100 км пробега автомобиля < 8 л:

коэффициент усиления усилителя на средних частотах > 300;

быстродействие процессора > 40 Мфлопс.

Классификация моделей и параметров, используемых при автоматизированном проектировании. В автоматизированных проектных процедурах вместо еще не существующего проектируемого объекта оперируют некоторым квазиобъектом — *моделью*, которая отражает некоторые интересующие исследователя свойства объекта. Модель может быть *физическим* объектом (макет, стенд) или *спецификацией.* Среди моделей-спецификаций различают упомянутые выше функциональные, поведенческие, информационные, структурные модели (описания). Эти модели называют *математическими,* если они формализованы средствами аппарата и языка математики.

В свою очередь, математические модели могут быть геометрическими, топологическими, динамическими, логическими и т.п., если они отражают соответствующие свойства объектов. Наряду с математическими моделями при проектировании используют рассматриваемые ниже функциональные IDEF0-модели, информационные модели в виде диаграмм сущность-отношение, геометрические модели-чертежи. В дальнейшем, если нет специальной оговорки, под словом “модель” будем подразумевать математическую модель.

Математическая функциональная модель в общем случае представляет собой алгоритм вычисления вектора выходных параметров **Y** при заданных векторах параметров элементов **X** и внешних параметров **Q**.

Математические модели могут быть символическими и численными. При использовании *символических* моделей оперируют не значениями величин, а их символическими обозначениями (идентификаторами). *Численные* модели могут быть *аналитическими*, т.е. их можно представить в виде явно выраженных зависимостей выходных параметров **Y** от параметров внутренних **X** и внешних **Q**, или *алгоритмическими*, в которых связь **Y, X** и **Q** задана неявно в виде алгоритма моделирования. Важнейший частный случай алгоритмических моделей — *имитационные*, они отображают процессы в системе при наличии внешних воздействий на систему. Другими словами, имитационная модель — это алгоритмическая поведенческая модель.

Классификацию математических моделей выполняют также по ряду других признаков.

Так, в зависимости от принадлежности к тому или иному иерархическому уровню выделяют модели уровней системного, функционально-логического, макроуровня (сосредоточенного) и микроуровня (распределенного).

По характеру используемого для описания математического аппарата различают модели лингвистические, теоретико-множественные, абстрактно-алгебраические, нечеткие, автоматные и т.п.

Например, на системном уровне преимущественно применяют модели систем массового обслуживания и сети Петри, на функционально-логическом уровне — автоматные модели на основе аппарата передаточных функций или конечных автоматов, на макроуровне — системы алгебро-дифференциальных уравнений, на микроуровне — дифференциальные уравнения в частных производных. Осо-

бое место занимают геометрические модели, используемые в системах конструирования.

Кроме того, введены понятия полных моделей и макромоделей, моделей статических и динамических, детерминированных и стохастических, аналоговых и дискретных, символических и численных.

Полная модель объекта в отличие от *макромодели* описывает не только процессы на внешних выводах моделируемого объекта, но и внутренние для объекта процессы.

Статические модели описывают статические состояния, в них не присутствует время в качестве независимой переменной. *Динамические* модели отражают поведение системы, т.е. в них обязательно используется время.

Стохастические и *детерминированные* модели различаются в зависимости от учета или неучета случайных факторов.

В *аналоговых* моделях фазовые переменные — непрерывные величины, в *дискретных* — дискретные, в частном случае дискретные модели являются *логическими* (*булевыми*), в них состояние системы и ее элементов описывается булевыми величинами. В ряде случаев полезно применение *смешанных* моделей, в которых одна часть подсистем характеризуется аналоговыми моделями, другая — логическими.

Информационные модели относятся к информационной страте автоматизированных систем, их используют прежде всего при инфологическом проектировании баз данных (БД) для описания связей между единицами информации.

Наибольшие трудности возникают при создании моделей слабоструктурированных систем, что характерно прежде всего для системного уровня проектирования. Здесь значительное внимание уделяется экспертным методам. В теории систем сформулированы общие рекомендации по подбору экспертов при разработке модели, организации экспертизы, по обработке полученных результатов. Достаточно общий подход к построению моделей сложных слабоструктурированных систем выражен в методиках IDEF.

Обычно в имитационных моделях фигурируют фазовые переменные. Так, на макроуровне имитационные модели представляют собой системы алгебро-дифференциальных уравнений

$$\mathbf{\Phi}(d\mathbf{V}/dt, \mathbf{V}, t) = 0, \qquad \text{при } t = 0 \;\; \mathbf{V} = \mathbf{V}_0, \tag{1.1}$$

где $\mathbf{V}$ — вектор фазовых переменных; t — время; $\mathbf{V}_0$ — вектор начальных условий. К примерам фазовых переменных можно отнести токи и напряжения в электрических системах, силы и скорости — в механических, давления и расходы — в гидравлических.

Выходные параметры систем могут быть двух типов. Во-первых, это *параметры-функционалы*, т.е. функционалы зависимостей $\mathbf{V}(t)$ в случае использования (1.1). Примеры таких параметров: амплитуды сигналов, временные задержки, мощности рассеивания и т.п. Во-вторых, это параметры, характеризующие способность проектируемого объекта работать при определенных внешних условиях. Эти выходные параметры являются граничными значениями диапазонов внешних переменных, в которых сохраняется работоспособность объекта.

Типовые проектные процедуры. Создать проект объекта (изделия или процесса) означает выбрать структуру объекта, определить значения всех его параметров и представить результаты в установленной форме. Результаты (проектная документация) могут быть выражены в виде чертежей, схем, пояснительных записок, программ для программно-управляемого технологического оборудования и других документов на бумаге или на машинных носителях информации.

Разработка (или выбор) структуры объекта есть проектная процедура, называемая *структурным синтезом,* а расчет (или выбор) значений параметров элементов $\mathbf{X}$ — процедура *параметрического синтеза*.

Задача структурного синтеза формулируется в системотехнике как *задача принятия решений* (ЗПР). Ее суть заключается в определении цели, множества возможных решений и ограничивающих условий.

Классификацию ЗПР осуществляют по ряду признаков. По числу критериев различают задачи одно- и многокритериальные. По степени неопределенности различают ЗПР детерминированные, ЗПР в условиях риска — при наличии в формулировке задачи случайных параметров, ЗПР в услови-

ях неопределенности, т.е. при неполноте или недостоверности исходной информации.

Реальные задачи проектирования, как правило, являются многокритериальными. Одна из основных проблем постановки многокритериальных задач — установление правил предпочтения вариантов. Способы сведения многокритериальных задач к однокритериальным и последующие пути решения изучаются в дисциплинах, посвященных методам оптимизации и математическому программированию.

Наличие случайных факторов усложняет решение ЗПР. Основные подходы к решению ЗПР в условиях риска заключаются или в решении "для наихудшего случая", или в учете в целевой функции математического ожидания и дисперсии выходных параметров. В первом случае задачу решают как детерминированную при завышенных требованиях к качеству решения, что является главным недостатком подхода. Во втором случае достоверность результатов решения намного выше, но возникают трудности с оценкой целевой функции. Применение метода Монте-Карло в случае алгоритмических моделей становится единственной альтернативой и, следовательно, для решения требуются значительные вычислительные ресурсы.

Существуют две группы ЗПР в условиях неопределенности. Одна из них решается при наличии противодействия разумного противника. Такие задачи изучаются в *теории игр*, для задач проектирования в технике они не характерны. Во второй группе достижению цели противодействие оказывают силы природы. Для их решения полезно использовать теорию и методы *нечетких множеств*.

Например, при синтезе структуры автоматизированной системы постановка задачи должна включать в качестве исходных данных следующие сведения:

— множество выполняемых системой функций (другими словами, множество работ, каждая из которых может состоять из одной или более операций); возможно, что в этом множестве имеется частичная упорядоченность работ, что может быть представлено в виде ориентированного графа, в котором вершины соответствуют работам, а дуги — отношениям порядка;

— типы допустимых для использования серверов (машин), выполняющих функции системы;

— множество внешних источников и потребителей информации;

— во многих случаях задается также некоторая исходная структура системы в виде взаимосвязанной совокупности серверов определенных типов; эта структура может рассматриваться как обобщенная избыточная или как вариант первого приближения;

—различного рода ограничения, в частности, ограничения на затраты материальных ресурсов и (или) на времена выполнения функций системы.

Задача заключается в синтезе (или коррекции) структуры, определении типов серверов (программно-аппаратных средств), распределении функций по серверам таким образом, чтобы достигался экстремум целевой функции при выполнении заданных ограничений.

Конструирование, разработка технологических процессов, оформление проектной документации — частные случаи структурного синтеза.

Задачу параметрического синтеза называют параметрической *оптимизацией* (или оптимизацией), если ее решают как задачу математического программирования

$$\mathbf{extr}\ F(\mathbf{X}),\ \mathbf{X} \in \mathbf{D}_x,$$

где $F(\mathbf{X})$ — целевая функция; $\mathbf{X}$ — вектор управляемых (называемых также проектными или варьируемыми) параметров; $\mathbf{D}_x = \{\mathbf{X}|\ \boldsymbol{\varphi}(\mathbf{X}) < 0,\ \boldsymbol{\psi}(\mathbf{X}) = 0\}$ — допустимая область; $\boldsymbol{\varphi}(\mathbf{X})$ и $\boldsymbol{\psi}(\mathbf{X})$ — функции-ограничения.

Пример. Электронный усилитель: управляемые параметры $\mathbf{X}$ = (параметры резисторов, конденсаторов, транзисторов); выходные параметры $\mathbf{Y}$ = (*f*в и *f*н — граничные частоты полосы пропускания; *K* — коэффициент усиления на средних частотах; *R*вх — входное сопротивление). В качестве целевой функции $F(\mathbf{X})$ можно выбрать параметр *f*в, а условия работоспособности остальных выходных параметров отнести к функциям-ограничениям.

Следующая после синтеза группа проектных процедур — процедуры анализа. Цель *анализа* — получение информации о характере функционирования и значениях выходных параметров **Y** при заданных структуре объекта, сведениях о внешних параметрах **Q** и параметрах элементов **X**. Если заданы фиксированные значения параметров **X** и **Q,** то имеет место процедура *одновариантного анализа*, которая сводится к решению уравнений математической модели, например, такой, как модель (1.1), и вычислению вектора выходных параметров **Y**. Если заданы статистические сведения о параметрах **X** и нужно получить оценки числовых характеристик распределений выходных параметров (например,

оценки математических ожиданий и дисперсий), то это процедура *статистического анализа*. Если требуется рассчитать матрицы абсолютной **A** и (или) относительной **B** чувствительности, то имеет место задача *анализа чувствительности*.

Элемент A_{ji} матрицы **A** называют *абсолютным коэффициентом чувствительности*, он представляет собой частную производную j-го выходного параметра y_j по i-ому параметру x_i. Другими словами, A_{ji} является элементом вектора градиента j-го выходного параметра. На практике удобнее использовать безразмерные *относительные коэффициенты чувствительности* B_{ji}, характеризующие степень влияния изменений параметров элементов на изменения выходных параметров:

$$B_{ji} = A_{ji}\, x_{i\text{ном}} / y_{j\text{ном}},$$

где $x_{i\text{ном}}$ и $y_{j\text{ном}}$ — номинальные значения параметров x_i и y_j соответственно.

В процедурах *многовариантного анализа* определяется влияние внешних параметров, разброса и нестабильности параметров элементов на выходные параметры. Процедуры статистического анализа и анализа чувствительности — характерные примеры процедур многовариантного анализа.

1.3. Системы автоматизированного проектирования и их место среди других автоматизированных систем.

Структура САПР. Как и любая сложная система, САПР состоит из подсистем (рис. 1.1). Различают подсистемы проектирующие и обслуживающие.

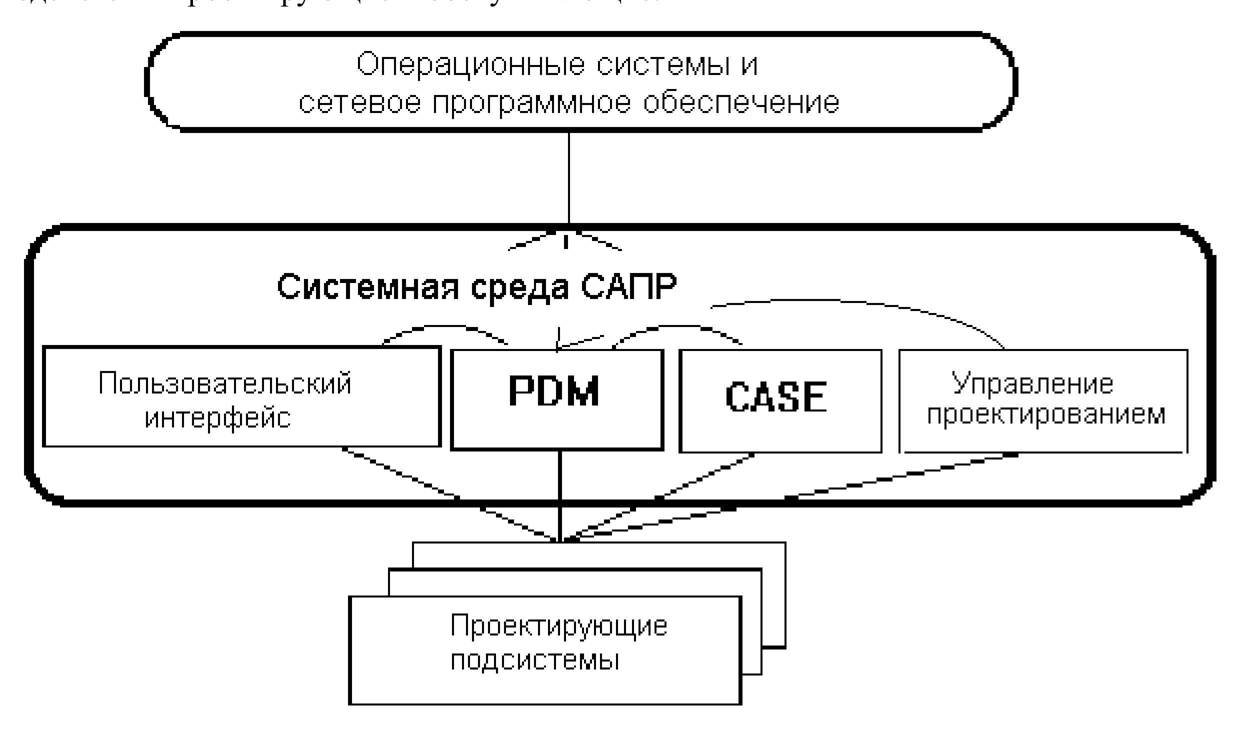


Рис 1.1. Структура программного обеспечения САПР

Проектирующие подсистемы непосредственно выполняют проектные процедуры. Примерами проектирующих подсистем могут служить подсистемы геометрического трехмерного моделирования механических объектов, изготовления конструкторской документации, схемотехнического анализа, трассировки соединений в печатных платах.

Обслуживающие подсистемы обеспечивают функционирование проектирующих подсистем, их совокупность часто называют системной средой (или оболочкой) САПР. Типичными обслуживающими подсистемами являются подсистемы управления проектными данными (PDM — Product Data Management), управления процессом проектирования (DesPM — Design Process Management), пользовательского интерфейса для связи разработчиков с ЭВМ, CASE (Computer Aided Software Engineering) для разработки и сопровождения программного обеспечения САПР, обучающие подсистемы для освоения пользователями технологий, реализованных в САПР.

Структурирование САПР по различным аспектам обусловливает появление *видов обеспечения* САПР. Принято выделять семь видов обеспечения:

— *техническое* (ТО), включающее различные аппаратные средства (ЭВМ, периферийные устройства, сетевое коммутационное оборудование, линии связи, измерительные средства);

— *математическое* (МО), объединяющее математические методы, модели и алгоритмы для выполнения проектирования;

— *программное* (ПО), представляемое компьютерными программами САПР;

— *информационное* (ИО), состоящее из баз данных (БД), систем управления базами данных (СУБД), а также других данных, используемых при проектировании; отметим, что вся совокупность используемых при проектировании данных называется информационным фондом САПР, а БД вместе с СУБД носит название банка данных (БнД);

— *лингвистическое* (ЛО), выражаемое языками общения между проектировщиками и ЭВМ, языками программирования и языками обмена данными между техническими средствами САПР;

— *методическое* (МетО), включающее различные методики проектирования, иногда к МетО относят также математическое обеспечение;

— *организационное* (ОО), представляемое штатными расписаниями, должностными инструкциями и другими документами, регламентирующими работу проектного предприятия.

Разновидности САПР. Классификацию САПР осуществляют по ряду признаков, например, по приложению, целевому назначению, масштабам (комплексности решаемых задач), характеру базовой подсистемы — ядра САПР.

По *приложениям* наиболее представительными и широко используемыми являются следующие группы САПР.

1. САПР для применения в отраслях общего машиностроения. Их часто называют машиностроительными САПР или MCAD (Mechanical CAD) системами.

2. САПР для радиоэлектроники. Их названия — ECAD (Electronic CAD) или EDA (Electronic Design Automation) системы.

3. САПР в области архитектуры и строительства.

Кроме того, известно большое число более специализированных САПР, или выделяемых в указанных группах, или представляющих самостоятельную ветвь в классификации. Примерами таких систем являются САПР больших интегральных схем (БИС); САПР летательных аппаратов; САПР электрических машин и т.п.

По *целевому назначению* различают САПР или подсистемы САПР, обеспечивающие разные аспекты (страты) проектирования. Так, в составе MCAD появляются CAE/CAD/CAM системы :

1. САПР функционального проектирования, иначе САПР-Ф или CAE (Computer Aided Engineering) системы.

2. *конструкторские* САПР общего машиностроения — САПР-К, часто называемые просто CAD системами;

3. *технологические* САПР общего машиностроения — САПР-Т, иначе называемые автоматизированными системами технологической подготовки производства АСТПП или системами CAM (Computer Aided Manufacturing).

По *масштабам* различают отдельные программно-методические комплексы (ПМК) САПР, например, комплекс анализа прочности механических изделий в соответствии с методом конечных элементов (МКЭ) или комплекс анализа электронных схем; системы ПМК; системы с уникальными архитектурами не только программного (software), но и технического (hardware) обеспечений.

По *характеру базовой подсистемы* различают следующие разновидности САПР.

1. САПР *на базе подсистемы машинной графики и геометрического моделирования*. Эти САПР ориентированы на приложения, где основной процедурой проектирования является конструирование, т.е. определение пространственных форм и взаимного расположения объектов. Поэтому к этой группе систем относится большинство графических ядер САПР в области машиностроения.

В настоящее время появились унифицированные графические ядра, применяемые более чем в одной САПР, это ядра Parasolid фирмы EDS Unigraphics и ACIS фирмы Intergraph.

2. САПР *на базе* СУБД. Они ориентированы на приложения, в которых при сравнительно несложных математических расчетах перерабатывается большой объем данных. Такие САПР преимущественно встречаются в технико-экономических приложениях, например, при проектировании бизнес-планов, но имеют место также при проектировании объектов, подобных щитам управления в системах автоматики.

3. САПР *на базе конкретного прикладного пакета*. Фактически это автономно используемые программно-методические комплексы, например, имитационного моделирования производственных процессов, расчета прочности по методу конечных элементов, синтеза и анализа систем автоматического управления и т.п. Часто такие САПР относятся к системам CAE. Примерами могут служить программы логического проектирования на базе языка VHDL, математические пакеты типа MathCAD.

4. *Комплексные* (*интегрированные)* САПР, состоящие из совокупности подсистем предыдущих видов. Характерными примерами комплексных САПР являются CAE/CAD/CAM-системы в машиностроении или САПР БИС. Так, САПР БИС включает в себя СУБД и подсистемы проектирования компонентов, принципиальных, логических и функциональных схем, топологии кристаллов, тестов для проверки годности изделий. Для управления столь сложными системами применяют специализированные *системные среды*.

Функции, характеристики и примеры CAE/CAD/CAM-систем. Функции CAD-систем в машиностроении подразделяют на функции двухмерного (2*D*) и трехмерного (3*D*) проектирования. К функциям 2*D* относятся черчение, оформление конструкторской документации; к функциям 3*D* — получение трехмерных моделей, метрические расчеты, реалистичная визуализация, взаимное преобразование 2*D* и 3*D* моделей.

Среди CAD-систем различают “легкие” и “тяжелые” системы. Первые из них ориентированы преимущественно на 2*D* графику, сравнительно дешевы и менее требовательны в отношении вычислительных ресурсов. Вторые ориентированы на геометрическое моделирование (3*D*), более универсальны, дороги, оформление чертежной документации в них обычно осуществляется с помощью предварительной разработки трехмерных геометрических моделей.

Основные функции CAM-систем: разработка технологических процессов, синтез управляющих программ для технологического оборудования с числовым программным управлением (ЧПУ), моделирование процессов обработки, в том числе построение траекторий относительного движения инструмента и заготовки в процессе обработки, генерация постпроцессоров для конкретных типов оборудования с ЧПУ (*NC* — Numerical Control), расчет норм времени обработки.

Наиболее известны (к 1999 г.) следующие CAE/CAD/CAM-системы, предназначенные для машиностроения. “Тяжелые” системы (в скобках указана фирма, разработавшая или распространяющая продукт): Unigraphics (EDS Unigraphics); Solid Edge (Intergraph); Pro/Engineer (PTC — Parametric Technology Corp.), CATIA (Dassault Systemes), EUCLID (Matra Datavision), CADDS.5 (Computervision, ныне входит в PTC) и др.

“Легкие” системы: AutoCAD (Autodesk); АДЕМ; bCAD (ПроПро Группа, Новосибирск); Caddy (Ziegler Informatics); Компас (Аскон, С.Петербург); Спрут (Sprut Technology, Набережные Челны); Кредо (НИВЦ АСК, Москва).

Системы, занимающие промежуточное положение (среднемасштабные): Cimatron, Microstation (Bentley), Euclid Prelude (Matra Datavision), T-FlexCAD (Топ Системы, Москва) и др. С ростом возможностей персональных ЭВМ грани между “тяжелыми” и “легкими” CAD/CAM-системами постепенно стираются.

Функции CAE-систем довольно разнообразны, так как связаны с проектными процедурами анализа, моделирования, оптимизации проектных решений. В состав машиностроительных CAE-систем прежде всего включают программы для следующих процедур:

— моделирование полей физических величин, в том числе анализ прочности, который чаще всего выполняется в соответствии с МКЭ;

— расчет состояний и переходных процессов на макроуровне;

— имитационное моделирование сложных производственных систем на основе моделей массового обслуживания и сетей Петри.

Примеры систем моделирования полей физических величин в соответствии с МКЭ: Nastran, Ansys, Cosmos, Nisa, Moldflow.

Примеры систем моделирования динамических процессов на макроуровне: Adams и Dyna — в механических системах, Spice — в электронных схемах, ПА9 — для многоаспектного моделирования, т.е. для моделирования систем, принципы действия которых основаны на взаимовлиянии физических процессов различной природы.

Для удобства адаптации САПР к нуждам конкретных приложений, для ее развития целесообразно иметь в составе САПР инструментальные средства адаптации и развития. Эти средства представлены той или иной CASE-технологией, включая языки расширения. В некоторых САПР применяют оригинальные инструментальные среды.

Примерами могут служить объектно-ориентированная интерактивная среда CAS.CADE в системе EUCLID, содержащая библиотеку компонентов, в САПР T-Flex CAD 3*D* предусмотрена разработка дополнений в средах Visual C++ и Visual Basic.

Важное значение для обеспечения открытости САПР, ее интегрируемости с другими автоматизированными системами (АС) имеют интерфейсы, представляемые реализованными в системе форматами межпрограммных обменов. Очевидно, что, в первую очередь, необходимо обеспечить связи между CAE, CAD и CAM-подсистемами.

В качестве языков — форматов межпрограммных обменов — используются IGES, DXF, Express (стандарт ISO 10303-11, входит в совокупность стандартов STEP), SAT (формат ядра ACIS) и др.

Наиболее перспективными считаются диалекты языка Express, что объясняется общим характером стандартов STEP, их направленностью на различные приложения, а также на использование в современных распределенных проектных и производственных системах. Действительно, такие форматы, как IGES или DXF, описывают только геометрию объектов, в то время как в обменах между различными САПР и их подсистемами фигурируют данные о различных свойствах и атрибутах изделий.

Язык Express используется во многих системах интерфейса между CAD/CAM-системами. В частности, в систему CAD++ STEP включена среда SDAI (Standard Data Access Interface), в которой возможно представление данных об объектах из разных систем CAD и приложений (но описанных по правилам языка Express). CAD++ STEP обеспечивает доступ к базам данных большинства известных САПР с представлением извлекаемых данных в виде STEP-файлов. Интерфейс программиста позволяет открывать и закрывать файлы проектов в базах данных, производить чтение и запись сущностей. В качестве объектов могут использоваться точки, кривые, поверхности, текст, примеры проектных решений, размеры, связи, типовые изображения, комплексы данных и т.п.

Понятие о CALS-технологии. CALS-технология — это технология комплексной компьютеризации сфер промышленного производства, цель которой — унификация и стандартизация спецификаций промышленной продукции на всех этапах ее жизненного цикла. Основные спецификации представлены проектной, технологической, производственной, маркетинговой, эксплуатационной документацией. В CALS-системах предусмотрены хранение, обработка и передача информации в компьютерных средах, оперативный доступ к данным в нужное время и в нужном месте. Соответствующие системы автоматизации назвали автоматизированными логистическими системами или CALS (Computer Aided Logistic Systems). Поскольку под логистикой обычно понимают дисциплину, посвященную вопросам снабжения и управления запасами, а функции CALS намного шире и связаны со всеми этапами жизненного цикла промышленных изделий, применяют и более соответствующую предмету расшифровку аббревиатуры CALS — Continuous Acquisition and LifeCycle Support.

Применение CALS позволяет существенно сократить объемы проектных работ, так как описания многих составных частей оборудования, машин и систем, проектировавшихся ранее, хранятся в базах данных сетевых серверов, доступных любому пользователю технологии CALS. Существенно облегчается решение проблем ремонтопригодности, интеграции продукции в различного рода системы и среды, адаптации к меняющимся условиям эксплуатации, специализации проектных организаций и т.п. Ожидается, что успех на рынке сложной технической продукции будет немыслим вне технологии CALS.

Развитие CALS-технологии должно привести к появлению так называемых *виртуальных производств*, при которых процесс создания спецификаций с информацией для программно управляемого технологического оборудования, достаточной для изготовления изделия, может быть распределен во времени и пространстве между многими организационно автономными проектными студиями. Среди несомненных достижений CALS-технологии следует отметить легкость распространения передовых проектных решений, возможность многократного воспроизведения частей проекта в новых разработках и др.

Построение открытых распределенных автоматизированных систем для проектирования и управления в промышленности составляет основу современной CALS-технологии. Главная проблема их построения — обеспечение единообразного описания и интерпретации данных, независимо от места

и времени их получения в общей системе, имеющей масштабы вплоть до глобальных. Структура проектной, технологической и эксплуатационной документации, языки ее представления должны быть стандартизованными. Тогда становится реальной успешная работа над общим проектом разных коллективов, разделенных во времени и пространстве и использующих разные CAE/CAD/CAM-системы. Одна и та же конструкторская документация может быть использована многократно в разных проектах, а одна и та же технологическая документация адаптирована к разным производственным условиям, что позволяет существенно сократить и удешевить общий цикл проектирования и производства. Кроме того, упрощается эксплуатация систем.

Следовательно, информационная интеграция является неотъемлемым свойством CALS-систем. Поэтому в основу CALS-технологии положен ряд стандартов, обеспечивающих такую интеграцию.

Важные проблемы, требующие решения при создании комплексных САПР — управление сложностью проектов и интеграция ПО. Эти проблемы включают вопросы декомпозиции проектов, распараллеливания проектных работ, целостности данных, межпрограммных интерфейсов и др.

Комплексные автоматизированные системы. Известно, что частичная автоматизация зачастую не дает ожидаемого повышения эффективности функционирования предприятий. Поэтому предпочтительным является внедрение интегрированных САПР, автоматизирующих все основные этапы проектирования изделий. Дальнейшее повышение эффективности производства и повышение конкурентоспособности выпускаемой продукции возможно за счет интеграции систем проектирования, управления и документооборота.

Такая интеграция лежит в основе создания *комплексных систем автоматизации*, в которых помимо функций собственно САПР реализуются средства для автоматизации функций управления проектированием, документооборота, планирования производства, учета и т.п.

Проблемы интеграции лежат в основе технологии Юпитер, пропагандируемой фирмой Intergraph. Пример сращивания некоторых подсистем из САПР и АСУ — программный продукт TechnoDOCS (российская фирма Весть). Его функции:

— интеграция программ документооборота с проектирующими пакетами (конкретно с AutoCAD, Microstation и другими программами, исполняемыми в Windows-средах и поддерживающими взаимодействие по технологиям DDE или OLE, разработанным фирмой Microsoft);

— ведение архива технической документации;

— маршрутизация работ и прохождение документации, контроль исполнения;

— управление параллельным проектированием, т.е. координацией проектных работ, выполняемых коллективно.

Очевидно, что подобная интеграция является неотъемлемой чертой CALS-систем. В основу CALS-технологии положен ряд стандартов и прежде всего это стандарты STEP, а также Parts Library, Mandate, SGML (Standard Generalized Markup Language), EDIFACT (Electronic Data Interchange For Administration, Commerse, Transport) и др. Стандарт SGML устанавливает способы унифицированного оформления документов определенного назначения — отчетов, каталогов, бюллетеней и т.п., а стандарт EDIFACT — способы обмена подобными документами.

Одна из наиболее известных реализаций CALS-технологии разработана фирмой Computervision. Это технология названа EPD (Electronic Product Definition) и ориентирована на поддержку процессов проектирования и эксплуатации изделий машиностроения.

В CALS-системах на всех этапах жизненного цикла изделий используется документация, полученная на этапе проектирования. Поэтому естественно, что составы подсистем в CALS и комплексных САПР в значительной мере совпадают.

Технологию EPD реализуют:

— CAD — система автоматизированного проектирования;

— CAM — автоматизированная система технологической подготовки производства (АСТПП);

— CAE — система моделирования и расчетов;

— CAPE (Concurrent Art-to-Product Environoment) — система поддержки параллельного проектирования (concurrent engineering);

— PDM — система управления проектными данными, представляющая собой специализированную СУБД (DBMS — Data Base Management System);

— 3*D* Viewer -система трехмерной визуализации;

— CADD — система документирования;

— CASE — система разработки и сопровождения программного обеспечения;

— методики обследования и анализа функционирования предприятий.

Основу EPD составляют системы CAD и PDM, в качестве которых используются CADDS5 и Optegra соответственно.

В значительной мере специфику EPD определяет система Optegra. В ней отображается иерархическая структура из-

делий, включающая все сборочные узлы и детали. В Optegra можно получить информацию об атрибутах любого элемента структуры, а также ответы на типичные для баз данных вопросы типа “Укажите детали из материала *X*” или “В каких блоках используются детали изготовителя *Y*?” и т.п.

Важной для пользователей особенностью Optegra является работа вместе с многооконной системой визуализации 3*D* Viewer. Пользователь может одновременно следить за информацией в нескольких типовых окнах:

— информационный браузер, в котором высвечиваются данные, запрашиваемые пользователем, например, из почтового ящика, Internet, корпоративных ресурсов, его персональной БД;

— окно структуры изделия, представляемой в виде дерева. Можно получать ответы на запросы подсветкой деталей D_j (листьев дерева), удовлетворяющих условиям запроса;

— 3*D* визуализатор, в этом окне высвечивается трехмерное изображение изделия, ответы на запросы даются и в этом окне цветовым выделением деталей D_j;

— окно пользовательского процесса, в котором в нужной последовательности в виде иконок отображается перечень задач, заданный пользователю для решения.

В системе Optegra связи между объектами задаются по протоколам стандартов STEP, внешний интерфейс осуществляется через базу данных SDAI.

Системы управления в составе комплексных автоматизированных систем. Системы управления в промышленности, как и любые сложные системы, имеют иерархическую структуру. Если рассматривать предприятие как систему верхнего уровня, то следующими уровнями по нисходящей линии будут уровни завода, цеха, производственного участка, производственного оборудования. Автоматизация управления реализуется с помощью автоматизированных систем управления (АСУ).

Среди АСУ различают *автоматизированные системы управления предприятием* (АСУП) и *автоматизированные системы управления технологическими процессами* (АСУТП). АСУП охватывает уровни от предприятия до цеха, АСУТП — от цеха и ниже (на уровне цеха могут быть средства и АСУП, и АСУТП).

В АСУП выделяют подсистемы, выполняющие определенные функции (рис. 1.2), типичными среди них являются:

— календарное планирование производства, потребностей в мощностях и материалах;

— оперативное управление производством;

— сетевое планирование проектов;

— управление проектированием изделий;

— учет и нормирование трудозатрат;

— учет основных фондов;

— управление финансами;

— управление запасами (складским хозяйством);

— управление снабжением (статистика закупок, контракты на закупку);

— маркетинг (статистика и анализ реализации, контракты на реализацию, прогноз, реклама).

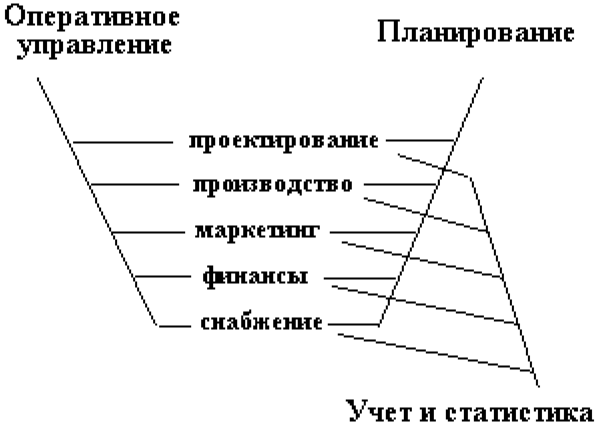


Рис. 1.2. Основные функции АСУП

Процедуры, выполняющие эти функции, часто называют *бизнес-функциями*, а маршруты решения задач управления, состоящие из бизнес-функций, называют *бизнес-процессами*.

Примечание. Как сказано выше, в САПР аналогичные понятия называют проектными процедурами и маршрутами проектирования.

Существуют разновидности АСУП со своими англоязычными названиями. Наиболее общую систему с перечисленными выше функциями называют ERP (Enterprise Resource Planning). Системы, направленные на управление информацией о материалах, производстве, контроле и т.п. изделий, называют MRP-2 (Manufacturing Resource Planning). В ERP, как и в САПР, важная роль отводится системам управления данными PDM. Если PDM обеспечивает управление конфигурацией проектов и относится в большей мере к проектированию, то MRP-2 управляет данными, относящимися к производству. Для таких систем иногда используют также название MES (Manufacturing Execution System).

Мировыми лидерами среди систем программного обеспечения АСУП являются системы R3 (фирма SAP) и Baan IV (Baan), широко известны также MANMAN/X (Computer Associates CIS), Elite Series (Tecsys Inc.), Mapix (IBM) и др. Примерами комплексных систем управления предприятием, созданных в России, служат системы АККОРД фирмы Атлант Информ, а также системы фирм Галактика и Парус. Корпоративные информационные системы разрабатывают также такие

российские фирмы, как АйТи, R-Style и др.

Характерные особенности современных АСУП.

1. Открытость по отношению к ведущим платформам (UNIX, Windows, OS/2) и различным СУБД и прежде всего мощным СУБД типа Oracle, Ingres, Informix, Sybase; поддержка технологий типа ODBC (Open Data Base Connection), OLE (Object Linking and Embedding), DDE (Dynamic Data Exchange); поддержка архитектур клиент/сервер. Важная характеристика — возможность работы в среде распределенных вычислений.

2. Возможность сквозного выполнения всех допустимых бизнес-функций или их части, что обеспечивается модульным построением (количество функций может превышать 100).

3. Адаптируемость к конкретным заказчикам и условиям рынка.

4. Наличие инструментальных средств, в том числе языка расширения или 4GL (языка четвертого поколения). Так, в R3 используется язык ABAP/L, в Elite Series — язык Informix-4GL.

5. Техническое обеспечение АСУП — компьютерная сеть, узлы которой расположены как в административных отделах предприятия, так и в цехах.

Очевидно, что для создания и развития виртуальных предприятий необходимы распространение CALS-технологии не только на САПР, но и на АСУ, их интеграция в комплексные системы информационной поддержки всех этапов жизненного цикла промышленной продукции.

Функциями АСУТП на уровнях цеха и участка являются сбор и обработка данных о состоянии оборудования и протекании производственных процессов для принятия решений по загрузке станков, по выполнению технологических маршрутов. Программное обеспечение АСУТП на этих уровнях представлено системой диспетчерского управления и сбора данных, называемой SCADA *(Supervisory Control and Data Acquisition*), а техническое обеспечение — персональными ЭВМ и микрокомпьютерами, связанными локальной вычислительной сетью. Кроме диспетчерских функций, SCADA выполняет роль инструментальной системы разработки программного обеспечения для промышленных систем компьютерной автоматизации, т.е. роль специфической CASE-системы. Для систем АСУТП характерно использование *программируемых контроллеров* (ПЛК или PLC — Programmed Logic Controller), — компьютеров, встроенных в технологическое оборудование.

Функции SCADA:

1. сбор первичной информации от датчиков;
2. хранение, обработка и визуализация данных;
3. управление и регистрация аварийных сигналов;
4. связь с корпоративной информационной сетью;
5. автоматизированная разработка прикладного ПО.

К разработке программ для программируемых контроллеров обычно привлекаются не профессиональные программисты, а заводские технологи. Поэтому языки программирования должны быть достаточно простыми, обычно построенными на визуальных изображениях ситуаций. Например, используются различные схемные языки. Ряд языков стандартизован и представлен в международном стандарте IEC 1131-3.

На уровне управления технологическим оборудованием в АСУТП выполняются запуск, тестирование, выключение станков, сигнализация о неисправностях, выработка управляющих воздействий для рабочих органов программно управляемого оборудования (*NC* — Numerical Control). Для этого в составе технологического оборудования используются системы управления на базе встроенных контроллеров.

Автоматизированные системы делопроизводства (АСД). Информационные технологии и автоматизированные системы управления документами и документооборотом пользуются все возрастающим вниманием среди предприятий и фирм различного профиля, поскольку организация работы с документами существенно влияет на эффективность производственных и бизнес-процессов. Такие системы имеют как самостоятельное значение, так и играют важную роль в интегрированных автоматизированных системах управления и проектирования.

Автоматизированные системы делопроизводства по своему назначению подразделяют на системы управления документами (СУД), управления документооборотом (СДО), управления знаниями

(в сфере делопроизводства) и инструментальные среды делопроизводства. В соответствии с другими критериями классификации системы делопроизводства подразделяют на специализированные и комплексные, локальные и распределенные, фактографические и документографические (полнотекстовые), заказные и тиражируемые.

Системы управления документами предназначены для обеспечения санкционированного доступа к документам. Характерные функции СУД:

— ввод документов, в частности, с помощью средств их автоматического распознавания;

— индексирование документов, например, оформление регистрационных карточек с полями для атрибутов; возможно атрибутивное индексирование — к атрибутам относятся автор документа, дата создания и ключевые слова или полнотекстовое индексирование — в индекс заносят весь текст, но без предлогов и окончаний некоторых слов.

— хранение документов;

— поиск нужных данных, который может быть атрибутивным в фактографических БД или полнотекстовым в случае слабоструктурированных документов;

— поддержка групповой работы над документами;

— разграничение прав доступа к документам;

— контроль и управление версиями документов, регламентирующие внесение в них изменений;

— сбор и анализ статистических данных по параметрам документов и функционированию системы;

— подготовка отчетов.

Системы управления документооборотом служат для управления деловыми процессами прохождения и обработки документов в соответствующих подразделениях и службах организации. Характерные функции СДО:

— регистрация документов при их вхождении в систему;

— маршрутизация документов, учет их движения (маршрутизация может быть жесткой при фиксированных маршрутах или свободной); управление потоками документов обеспечивает прохождение документов по заданному маршруту с контролем внесения в них резолюций, управление внесением изменений включает систему приоритетов, средства протоколирования изменений;

— контроль исполнения предписываемых документами действий;

— защита информации при ее передаче между пунктами распределенной системы;

— автоматическое уведомление соответствующих лиц о состоянии документов и содержащихся в них директив и рекомендаций;

— планирование работ, связанных с прохождением документов.

К *системам управления знаниями* в области делопроизводства относят системы, выполняющие функции, характерные для интеллектуальных систем. Примеры таких функций:

— классификация документов по тем или иным признакам;

— взаимное связывание документов, например, с помощью гипертекста;

— тематический отбор документов;

— интеграция данных, поступающих из различных источников;

— аналитическая обработка данных;

— моделирование деловых процессов.

Инструментальные среды в системах делопроизводства служат для формирования систем делопроизводства, адаптированных к условиям конкретных предприятий и фирм. Часто такое формирование производится путем дополнения некоторого базового компонента, в состав системы входит соответствующий язык расширения.

Кроме перечня решаемых задач, выделяют следующие свойства и характеристики систем делопроизводства:

— открытость, программные интерфейсы и форматы данных для обмена с другими информационными системами;

— мобильность для инсталляции на ведущих платформах;

— модульное построение, что обеспечивает масштабируемость — возможность эволюционного развития, адаптируемость, возможность внедрения на предприятиях по частям.

— пользовательский интерфейс;
— быстродействие, временные затраты на выполнение задач;
— уровень защиты информации;
— соответствие стандартам информационных технологий;
— операционные среды и используемые СУБД, требования к аппаратным ресурсам.
— перенос документов по мере их устаревания на более дешевые носители.

В крупных АСД предусматривается распределенное хранение с доступом в режимах как off-line, так и on-line. В первом случае пользователь формирует запрос в виде совокупности ключевых слов и направляет его средствами электронной почты (E-mail), СДО выдает список релевантных документов, пользователь выбирает из списка нужные документы и посылает вторичный более конкретный запрос, получая по E-mail запрошенные документы. Во втором случае используется связь в реальном времени, документ вызывается на экран компьютера и пользователь может непосредственно его просматривать и редактировать.

Современные корпоративные системы делопроизводства являются распределенными, имеющими архитектуру клиент-сервер. На серверной стороне находят применение серверы баз данных, полнотекстовых документов, электронной почты, приложений, SQL- и Web-серверы. На клиентской стороне могут выделяться рабочие места пользователей, администратора и разработчиков баз данных, информационно-поисковых систем, форм документов и т.п. В частности, применяются трехзвенные распределенные системы.

К широко известным системам документооборота и делопроизводства относятся Lotus Notes, Docs Open, ДЕЛО-96 и др. Преимущественно используемой ОС является Windows NT.

Упражнения и вопросы для самоконтроля

1. Дайте определение понятия “проектирование”.
2. Что является предметом изучения в теории систем?
3. Назовите признаки, присущие сложной системе.
4. Приведите примеры иерархической структуры технических объектов, их внутренних, внешних и выходных параметров.
5. Приведите примеры условий работоспособности.
6. Почему проектирование обычно имеет итерационный характер?
7. Какие причины привели к появлению и развитию CALS-технологии?
8. Приведите примеры проектных процедур, выполняемых в системах CAE, CAD, CAM.
9. Что понимают под комплексной автоматизированной системой?
10. Назовите основные типы промышленных автоматизированных систем и виды их обеспечения.
11. Назовите основные функции автоматизированных систем: САПР, АСУП, АСУТП, АСД.

Глава 2 Техническое обеспечение САПР

2.1. Структура ТО САПР

Требования к ТО САПР. Техническое обеспечение САПР включает в себя различные технические средства (hardware), используемые для выполнения автоматизированного проектирования, а именно ЭВМ, периферийные устройства, сетевое оборудование, а также оборудование некоторых вспомогательных систем (например, измерительных), поддерживающих проектирование.

Используемые в САПР технические средства должны обеспечивать:

1. выполнение всех необходимых проектных процедур, для которых имеется соответствующее ПО;
2. взаимодействие между проектировщиками и ЭВМ, поддержку интерактивного режима работы;
3. взаимодействие между членами коллектива, выполняющими работу над общим проектом.

Первое из этих требований выполняется при наличии в САПР вычислительных машин и систем с достаточными производительностью и емкостью памяти.

Второе требование относится к пользовательскому интерфейсу и выполняется за счет включения в САПР удобных средств ввода-вывода данных и прежде всего устройств обмена графической информацией.

Третье требование обусловливает объединение аппаратных средств САПР в *вычислительную сеть*.

В результате общая структура ТО САПР представляет собой сеть узлов, связанных между собой средой передачи данных (рис. 2.1). *Узлами* (станциями данных) являются рабочие места проектировщиков, часто называемые *автоматизированными рабочими местами* (АРМ) или *рабочими станциями* (WS — Workstation), ими могут быть также большие ЭВМ (мейнфреймы), отдельные периферийные и измерительные устройства. Именно в АРМ должны быть средства для интерфейса проектировщика с ЭВМ. Что касается вычислительной мощности, то она может быть распределена между различными узлами вычислительной сети.



Рис. 2.1. Структура технического обеспечения САПР

Среда передачи данных представлена каналами передачи данных, состоящими из линий связи и коммутационного оборудования.

В каждом узле можно выделить *оконечное оборудование данных* (ООД), выполняющее определенную работу по проектированию, и *аппаратуру окончания канала данных* (АКД), предназначенную для связи ООД со средой передачи данных. Например, в качестве ООД можно рассматривать персональный компьютер, а в качестве АКД — вставляемую в компьютер сетевую плату.

Канал передачи данных — средство двустороннего обмена данными, включающее в себя АКД и линию связи. *Линией связи* называют часть физической среды, используемую для распространения сигналов в определенном направлении, примерами линий связи могут служить коаксиальный кабель, витая пара проводов, волоконно-оптическая линия связи (ВОЛС). Близким является понятие *канала (канала связи*), под которым понимают средство односторонней передачи данных. Примером канала связи может быть полоса частот, выделенная одному передатчику при радиосвязи. В некоторой линии можно образовать несколько каналов связи, по каждому из которых передается своя информация. При этом говорят, что линия разделяется между несколькими каналами.

Типы сетей. Существуют два метода разделения линии передачи данных: *временное мультиплексирование* (иначе разделение по времени или TDM — Time Division Method), при котором каждому каналу выделяется некоторый квант времени, и *частотное разделение* (FDM — Frequency Division Method), при котором каналу выделяется некоторая полоса частот.

В САПР небольших проектных организаций, насчитывающих не более единиц-десятков ком-

пьютеров, которые размещены на малых расстояниях один от другого (например, в одной или нескольких соседних комнатах) объединяющая компьютеры сеть является локальной. *Локальная вычислительная сеть* (ЛВС или LAN — Local Area Network) имеет линию связи, к которой подключаются все узлы сети. При этом топология соединений узлов (рис. 2.2) может быть шинная (bus), кольцевая (ring), звездная (star). Протяженность линии и число подключаемых узлов в ЛВС ограничены.

В более крупных по масштабам проектных организациях в сеть включены десятки-сотни и более компьютеров, относящихся к разным проектным и управленческим подразделениям и размещенных в помещениях одного или нескольких зданий. Такую сеть называют *корпоративной*. В ее структуре можно выделить ряд ЛВС, называемых *подсетями*, и средства связи ЛВС между собой. В эти средства входят коммутационные серверы (блоки взаимодействия подсетей). Если коммутационные серверы объединены отделенными от ЛВС подразделений каналами передачи данных, то они образуют новую подсеть, называемую *опорной* (или транспортной), а вся сеть оказывается иерархической структуры.

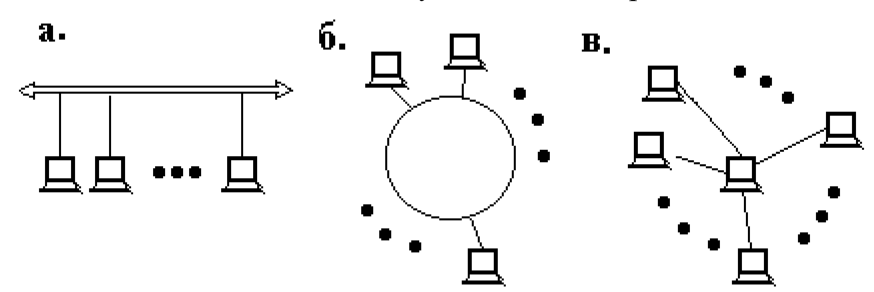


Рис. 2.2. Варианты топологии локальных вычислительных сетей: **а)** шинная; **б)** кольцевая; **в)** звездная

Если здания проектной организации удалены друг от друга на значительные расстояния (вплоть до их расположения в разных городах), то корпоративная сеть по своим масштабам становится *территориальной сетью* (WAN — Wide Area Network). В территориальной сети различают *магистральные* каналы передачи данных (магистральную сеть), имеющие значительную протяженность, и каналы передачи данных, связывающие ЛВС (или совокупность ЛВС отдельного здания или кампуса) с магистральной сетью и называемые *абонентской линией* или соединением "*последней мили*".

Обычно создание *выделенной* магистральной сети, т.е. сети, обслуживающей единственную организацию, обходится для нее слишком дорого. Поэтому чаще прибегают к услугам провайдера, т.е. организации, предоставляющей телекоммуникационные услуги многим пользователям. В этом случае внутри корпоративной сети связь на значительных расстояниях осуществляется через *магистральную сеть общего пользования*. В качестве такой сети можно использовать, например, городскую или междугородную телефонную сеть или территориальные сети передачи данных. Наиболее распространенной формой доступа к этим сетям в настоящее время является обращение к глобальной вычислительной сети Internet.

Для многих корпоративных сетей возможность выхода в Internet является желательной не только для обеспечения взаимосвязи удаленных сотрудников собственной организации, но и для получения других информационных услуг. Развитие виртуальных предприятий, работающих на основе CALS-технологий, с необходимостью подразумевает информационные обмены через территориальные сети, как правило, через Internet.

Структура ТО САПР для крупной организации представлена на рис. 2.3. Здесь показана типичная структура крупных корпоративных сетей САПР, называемая архитектурой *клиент-сервер*. В сетях клиент-сервер выделяется один или несколько узлов, называемых *серверами*, которые выполняют в сети управляющие или общие для многих пользователей проектные функции, а остальные узлы (рабочие места) являются терминальными, их называют *клиен-*

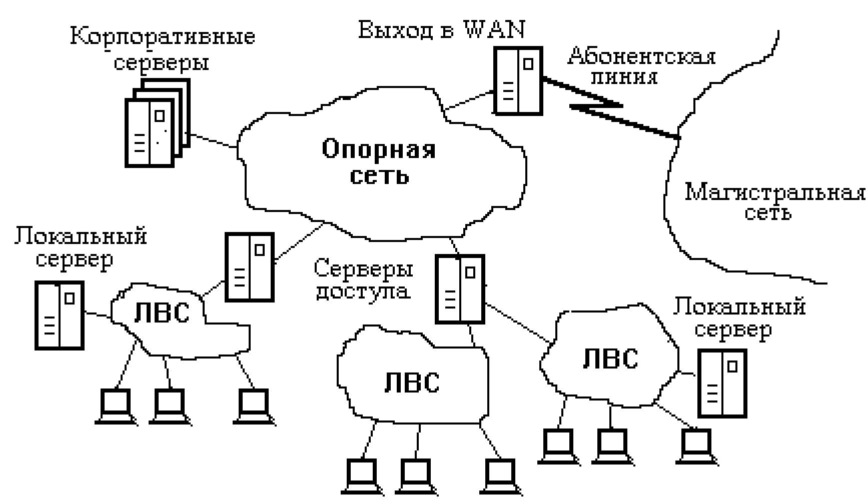


Рис. 2.3. Структура корпоративной сети САПР

тами, в них работают пользователи. В общем случае сервером называют совокупность программных средств, ориентированных на выполнение определенных функций, но если эти средства сосредоточены на конкретном узле вычислительной сети, то тогда понятие сервер относится именно к узлу сети.

Сети клиент-сервер различают по характеру распределения функций между серверами, другими словами, их классифицируют по типам серверов. Различают *файл-серверы* для хранения файлов, разделяемых многими пользователями, *серверы баз данных* автоматизированной системы, *серверы приложений* для решения конкретных прикладных задач, *коммутационные серверы* (называемые также блоками взаимодействия сетей или серверами доступа) для взаимосвязи сетей и подсетей, *специализированные серверы* для выполнения определенных телекоммуникационных услуг, например, серверы электронной почты.

В случае специализации серверов по определенным приложениям сеть называют *сетью распределенных вычислений*. Если сервер приложений обслуживает пользователей одной ЛВС, то естественно назвать такой сервер локальным. Но поскольку в САПР имеются приложения и базы данных, разделяемые пользователями разных подразделений и, следовательно, клиентами разных ЛВС, то соответствующие серверы относят к группе корпоративных, подключаемых обычно к опорной сети (см. рис. 2.3.).

Наряду с архитектурой клиент-сервер применяют о*дноранговые* сети, в которых любой узел в зависимости от решаемой задачи может выполнять как функции сервера, так и функции клиента. Организация взаимодействия в таких сетях при числе узлов более нескольких десятков становится чрезмерно сложной, поэтому одноранговые сети применяют только в небольших по масштабам САПР.

В соответствии со способами коммутации различают сети с *коммутацией каналов* и *коммутацией пакетов*. В первом случае при обмене данными между узлами *A* и *B* в сети создается физическое соединение между *A* и *B*, которое во время сеанса связи используется только этими абонентами. Примером сети с коммутацией каналов может служить телефонная сеть. Здесь передача информации происходит быстро, но каналы связи используются неэффективно, так как при обмене данными возможны длительные паузы и канал “простаивает.” При коммутации пакетов физического соединения, которое в каждый момент сеанса связи соединяло бы абонентов *A* и *B*, не создается. Сообщения разделяются на порции, называемые *пакетами*, которые передаются в разветвленной сети от *A* к *B* или обратно через промежуточные узлы с возможной буферизацией (временным запоминанием) в них. Таким образом, любая линия может разделяться многими сообщениями, попеременно пропуская при этом пакеты разных сообщений с максимальным заполнением упомянутых пауз.

Эталонная модель взаимосвязи открытых систем (ЭМВОС). Для удобства модернизации сложных информационных систем их делают максимально *открытыми*, т.е. приспособленными для внесения изменений в некоторую часть системы при сохранении неизменными остальных частей. В отношении вычислительных сетей реализация концепции открытости привела к появлению ЭМВОС, предложенной международной организацией стандартизации (ISO — International Standard Organization). В этой модели дано описание общих принципов, правил, соглашений, обеспечивающих взаимодействие информационных систем и называемых *протоколами*.

В ЭМВОС информационную сеть рассматривают как совокупность функций (протоколов), которые подразделяют на группы, называемые *уровнями*. Именно разделение на уровни позволяет вносить изменения в средства реализации одного уровня без перестройки средств других уровней, что значительно упрощает и удешевляет модернизацию средств по мере развития техники.

ЭМВОС содержит семь уровней.

На *физическом (physical)* уровне осуществляется представление информации в виде электрических или оптических сигналов, преобразование формы сигналов, выбор параметров физических сред передачи данных, организуется передача информации через физические среды.

На *канальном (link*) уровне выполняется обмен данными между соседними узлами сети, т.е. узлами, непосредственно связанными физическими соединениями без других промежуточных узлов. Отметим, что пакеты канального уровня обычно называют *кадрами*.

На *сетевом (network)* уровне происходит формирование пакетов по правилам тех промежуточных сетей, через которые проходит исходный пакет, и *маршрутизация* пакетов, т.е. определение и ре-

ализация маршрутов, по которым передаются пакеты. Другими словами, маршрутизация сводится к образованию логических каналов. *Логическим каналом* называют виртуальное соединение двух или более объектов сетевого уровня, при котором возможен обмен данными между этими объектами. Понятию логического канала необязательно соответствует физическое соединение линий передачи данных между связываемыми пунктами. Это понятие введено для абстрагирования от физической реализации соединения. Еще одной важной функцией сетевого уровня после маршрутизации является контроль нагрузки на сеть с целью предотвращения перегрузок, отрицательно влияющих на работу сети.

На *транспортном (transport)* уровне обеспечивается связь между оконечными пунктами (в отличие от предыдущего сетевого уровня, на котором обеспечивается передача данных через промежуточные компоненты сети). К функциям транспортного уровня относятся мультиплексирование и демультиплексирование (сборка-разборка пакетов в конечных пунктах), обнаружение и устранение ошибок в переданных данных, реализация заказанного уровня услуг (например, заказанных скорости и надежности передачи).

На *сеансовом (session)* уровне определяются тип связи (дуплекс или полудуплекс), начало и окончание заданий, последовательность и режим обмена запросами и ответами взаимодействующих партнеров.

На *представительном (presentation)* уровне реализуются функции представления данных (кодирование, форматирование, структурирование). Например, на этом уровне выделенные для передачи данные преобразуются из одного кода в другой.

На *прикладном (application)* уровне определяются и оформляются в сообщения те данные, которые подлежат передаче по сети.

В конкретных случаях может возникать потребность в реализации лишь части названных функций, тогда соответственно сеть будет содержать лишь часть уровней. Так, в простых (неразветвленных) ЛВС отпадает необходимость в средствах сетевого и транспортного уровней. Одновременно сложность функций канального уровня делает целесообразным его разделение в ЛВС на два подуровня: *управление доступом к каналу* (MAC — Medium Access Control) и *управление логическим каналом* *(*LLC — Logical Link Control). К подуровню LLC в отличие от подуровня MAC относится часть функций канального уровня, не зависящих от особенностей передающей среды.

Передача данных через разветвленные сети происходит при использовании *инкапсуляции-декапсуляции* порций данных. Так, сообщение, пришедшее на транспортный уровень, делится на сегменты, которые получают заголовки и передаются на сетевой уровень. Сегментом обычно называют пакет транспортного уровня. Сетевой уровень организует передачу данных через промежуточные сети. Для этого сегмент может быть разделен на части (пакеты), если сеть не поддерживает передачу сегментов целиком. Пакет снабжается своим сетевым заголовком (т.е. происходит инкапсуляция). При передаче между узлами промежуточной ЛВС требуется инкапсуляция пакетов в кадры с возможной разбивкой пакета. Приемник декапсулирует сегменты и восстанавливает исходное сообщение.

2.2. Аппаратура рабочих мест в автоматизированных системах проектирования и управления

Вычислительные системы в САПР. В качестве средств обработки данных в современных САПР широко используют рабочие станции, серверы, персональные компьютеры. Большие ЭВМ и в том числе суперЭВМ обычно не применяют, так как они дороги и их отношение производительность/цена существенно ниже подобного показателя серверов и многих рабочих станций.
На базе рабочих станций или персональных компьютеров создают АРМ.

Типичный состав устройств АРМ: ЭВМ с одним или несколькими микропроцессорами, оперативной и кэш-памятью и шинами, служащими для взаимной связи устройств; устройства ввода-вывода, включающие в себя, как минимум, клавиатуру, мышь, дисплей; дополнительно в состав АРМ могут входить принтер, сканер, плоттер (графопостроитель), дигитайзер и некоторые другие периферийные устройства.

Память ЭВМ обычно имеет иерархическую структуру. Поскольку в памяти большого объема трудно добиться одновременно высокой скорости записи и считывания данных, память делят на

сверхбыстродействующую кэш-память малой емкости, основную оперативную память умеренного объема и сравнительно медленную внешнюю память большой емкости, причем, в свою очередь, кэш-память часто разделяют на кэш первого и второго уровней.

Например, в персональных компьютерах на процессорах Pentium III кэш первого уровня имеет по 16 Кбайт для данных и для адресов, он и кэш второго уровня емкостью 256 Кбайт встроены в процессорный кристалл, емкость оперативной памяти составляет десятки-сотни Мбайт.

Для связи наиболее быстродействующих устройств (процессора, оперативной и кэш-памяти, видеокарты) используется системная шина с пропускной способностью до одного-двух Гбайт/с. Кроме системной шины на материнской плате компьютера имеются шина расширения для подключения сетевого контроллера и быстрых внешних устройств (например, шина PCI с пропускной способностью 133 Мбайт/с) и шина медленных внешних устройств, таких как клавиатура, мышь, принтер и т.п.

Рабочие станции (*workstation*) по сравнению с персональными компьютерами представляют собой вычислительную систему, специализированную на выполнение определенных функций. Специализация обеспечивается как набором программ, так и аппаратно за счет использования дополнительных специализированных процессоров. Так, в САПР для машиностроения преимущественно применяют графические рабочие станции для выполнения процедур геометрического моделирования и машинной графики. Эта направленность требует мощного процессора, высокоскоростной шины, памяти достаточно большой емкости.

Высокая производительность процессора необходима по той причине, что графические операции (например, перемещения изображений, их повороты, удаление скрытых линий и др.) часто выполняются по отношению ко всем элементам изображения. Такими элементами в трехмерной (3*D*) графике при аппроксимации поверхностей полигональными сетками являются многоугольники, их число может превышать 10^4. С другой стороны, для удобства работы проектировщика в интерактивном режиме задержка при выполнении команд указанных выше операций не должна превышать нескольких секунд. Но поскольку каждая такая операция по отношению к каждому многоугольнику реализуется большим числом машинных команд требуемое быстродействие составляет десятки миллионов машинных операций в секунду. Такое быстродействие при приемлемой цене достигается применением наряду с основным универсальным процессором также дополнительных специализированных (графических) процессоров, в которых определенные графические операции реализуются аппаратно.

В наиболее мощных рабочих станциях в качестве основных обычно используют высокопроизводительные микропроцессоры с сокращенной системой команд (с RISC-архитектурой), работающие под управлением одной из разновидностей операционной системы Unix. В менее мощных все чаще используют технологию Wintel (т.е. микропроцессоры Intel и операционные системы Windows). Графические процессоры выполняют такие операции, как, например, растеризация — представление изображения в растровой форме для ее визуализации, перемещение, вращение, масштабирование, удаление скрытых линий и т.п.

Типичные характеристики рабочих станций: несколько процессоров, десятки-сотни мегабайт оперативной и тысячи мегабайт внешней памяти, наличие кэш-памяти, системная шина со скоростями от сотен Мбайт/с до 1-2 Гбайт/с.

В зависимости от назначения существуют АРМ конструктора, АРМ технолога, АРМ руководителя проекта и т.п. Они могут различаться составом периферийных устройств, характеристиками ЭВМ.

В АРМ конструктора (графических рабочих станциях) используются растровые мониторы с цветными трубками. Типичные значения характеристик мониторов находятся в следующих пределах: размер экрана по диагонали 17…24 дюйма (фактически изображение занимает площадь на 5…8 % меньше, чем указывается в паспортных данных). Разрешающая способность монитора, т.е. число различимых пикселей (отдельных точек, из которых состоит изображение), определяется шагом между отверстиями в маске, через которые проходит к экрану электронный луч в электронно-лучевой трубке. Этот шаг находится в пределах 0,21…0,28 мм, что соответствует количеству пикселей изображения от 800×600 до 1920×1200 и более. Чем выше разрешающая способность, тем шире должна быть полоса пропускания электронных блоков видеосистемы при одинаковой частоте кадровой развертки. Полоса пропускания видеоусилителя находится в пределах 110…150 МГц и потому частота кадровой развертки обычно снижается с 135 Гц для разрешения 640×480 до 60 Гц для разрешения 1600×1200.

Отметим, что чем ниже частота кадровой развертки, а это есть частота регенерации изображения, тем заметнее мерцание экрана. Желательно, чтобы эта частота была не ниже 75 Гц.

Специально выпускаемые ЭВМ как серверы высокой производительности обычно имеют структуру симметричной многопроцессорной вычислительной системы. В них системная память разделяется всеми процессорами, каждый процессор может иметь свою сверхоперативную память сравнительно небольшой емкости, число процессоров невелико (единицы, редко более десяти). Например, сервер Enterprise 250 (Sun Microsystems) имеет 1-2 процессора, его цена в зависимости от комплектации колеблется в диапазоне 24-56 тыс. долларов, а сервер Enterprise 450 с четырьмя процессорами стоит от 82 до 95 тысяч долларов.

Периферийные устройства. Для ввода графической информации с имеющихся документов в САПР используют дигитайзеры и сканеры.

Дигитайзер применяют для ручного ввода. Он имеет вид кульмана, по его электронной доске перемещается курсор, на котором расположен визир и кнопочная панель. Курсор имеет электромагнитную связь с сеткой проводников в электронной доске. При нажатии кнопки в некоторой позиции курсора происходит занесение в память информации о координатах этой позиции. Таким образом может осуществляться ручная “сколка” чертежей.

Для автоматического ввода информации с имеющихся текстовых или графических документов используют *сканеры* планшетного или протяжного типа. Способ считывания — оптический. В сканирующей головке размещаются оптоволоконные самофокусирующиеся линзы и фотоэлементы. Разрешающая способность в разных моделях составляет от 300 до 800 точек на дюйм (этот параметр часто обозначают dpi). Считанная информация имеет растровую форму, программное обеспечение сканера представляет ее в одном из стандартных форматов, например TIFF, GIF, PCX, JPEG, и для дальнейшей обработки может выполнить векторизацию — перевод графической информации в векторную форму, например, в формат DXF.

Для вывода информации применяют принтеры и плоттеры. Первые из них ориентированы на получение документов малого формата (А3, А4), вторые — для вывода графической информации на широкоформатные носители.

В этих устройствах преимущественно используется растровый (т.е. построчный) способ вывода со струйной технологией печати. Печатающая система в струйных устройствах включает в себя картридж и головку. Картридж — баллон, заполненный чернилами (в цветных устройствах имеется несколько картриджей, каждый с чернилами своего цвета). Головка — матрица из сопел, из которых мельчайшие чернильные капли поступают на носитель. Физический принцип действия головки термический или пьезоэлектрический. При термопечати выбрасывание капель из сопла происходит под действием его нагревания, что вызывает образование пара и выбрасывание капелек под давлением. При пьезоэлектрическом способе пропускание тока через пьезоэлемент приводит к изменению размера сопла и выбрасыванию капли чернил. Второй способ дороже, но позволяет получить более высококачественное изображение.

Типичная разрешающая способность принтеров и плоттеров 300 dpi, в настоящее время она повышена до 720 dpi. В современных устройствах управление осуществляется встроенными микропроцессорами. Типичное время вывода монохромного изображения формата А1 находится в пределах от 2 до 7 мин, цветного — в два раза больше.

Дигитайзеры, сканеры, принтеры, плоттеры могут входить в состав АРМ или разделяться пользователями нескольких рабочих станций в составе локальной вычислительной сети.

Особенности технических средств в АСУТП. Специфические требования предъявляют к вычислительной аппаратуре, работающей в составе АСУТП в цеховых условиях. Здесь используют как обычные персональные компьютеры, так и специализированные программируемые логические контроллеры (ПЛК), называемые *промышленными компьютерами*. Специфика ПЛК — наличие нескольких аналоговых и цифровых портов, встроенный интерпретатор специализированного языка, детерминированные задержки при обработке сигналов, требующих незамедлительного реагирования. Однако ПЛК в отличие от персональных компьютеров IBM PC рассчитаны на решение ограниченного круга задач в силу специализированности программного обеспечения.

В целом промышленные компьютеры имеют следующие особенности: 1) работа в режиме реального времени (для промышленных персональных компьютеров разработаны такие ОС реального времени, как OS-9, QNX, VRTX и др.); 2) конструкция, приспособленная для работы ЭВМ в цеховых условиях (повышенные вибрации, электромагнитные помехи, запыленность, перепады температур, иногда взрывоопасность); 3) возможность встраивания дополнительных блоков управляющей, регистрирующей, сопрягающей аппаратуры, что помимо специальных конструкторских решений обеспечивается использованием стандартных шин и увеличением числа плат расширения; 4) автоматический перезапуск компьютера в случае “зависания” программы; 5) повышенные требования к надежности функционирования. В значительной мере специализация промышленных компьютеров определяется программным обеспечением. Конструктивно промышленный компьютер представляет собой корзину (крейт) с несколькими гнездами (слотами) для встраиваемых плат. Возможно использование мостов между крейтами. В качестве стандартных шин в настоящее время преимущественно используются шины VME-bus (Versabus Module Europe-bus) и PCI (Peripheral Component Interconnect).

VME-bus — системная шина для создания распределенных систем управления на основе встраиваемого оборудования (процессоры, накопители, контроллеры ввода-вывода). Представляет собой расширение локальной шины компьютера на несколько гнезд объединительной платы (до 21 слота), возможно построение многомастерных систем, т.е. систем, в которых ведущими могут быть два или более устройств. Имеет 32-разрядные немультиплексируемые шины данных и адресов, возможно использование мультиплексируемой 64-разрядной шины. Пропускная способность шины 320 Мбайт/с.

PCI — более удобная шина для однопроцессорных архитектур, получает все большее распространение. Пропускная способность до 264 Мбайт/с, разрядность шины 2x32 и (или) при мультиплексировании 64, архитектура с одним ведущим устройством. Имеется ряд разновидностей шины, например шина CompactPCI, в которой унифицирован ряд геометрических и механических параметров.

Программная связь с аппаратурой нижнего уровня (датчиками, исполнительными устройствами) происходит через драйверы. Межпрограммные связи реализуются через интерфейсы, подобные OLE. Для упрощения создания систем разработан стандарт OPC (OLE for Process Control).

2.3. Методы доступа в локальных вычислительных сетях

Множественный доступ с контролем несущей и обнаружением конфликтов. Типичная среда передачи данных в ЛВС — отрезок (сегмент) коаксиального кабеля. К нему через аппаратуру окончания канала данных подключаются узлы — компьютеры и, возможно, общее периферийное оборудование. Поскольку среда передачи данных общая, а запросы на сетевые обмены в узлах появляются асинхронно, то возникает проблема разделения общей среды между многими узлами, другими словами, проблема обеспечения доступа к сети.

Доступом к сети называют взаимодействие станции (узла сети) со средой передачи данных для обмена информацией с другими станциями. Управление доступом к сети — это установление последовательности, в которой станции получают доступ к среде передачи данных.

Различают случайные и детерминированные методы доступа. Среди случайных методов наиболее известен метод *множественного доступа с контролем несущей и обнаружением конфликтов* (МДКН/ОК). Англоязычное название метода — Carrier Sense Multiple Access / Collision Detection (CSMA/CD). Этот метод основан на контроле наличия электрических колебаний (несущей) в линии передачи данных и устранении конфликтов, возникающих в случае попыток одновременного начала передачи двумя или более станциями, путем повторения попыток захвата линии через случайный отрезок времени.

МДКН/ОК является широковещательным (broadcasting) методом. Все станции при применении МДКН/ОК равноправны по доступу к сети. Если линия передачи данных свободна, то в ней отсутствуют электрические колебания, что легко распознается любой станцией, желающей начать передачу. Такая станция захватывает линию. Любая другая станция, желающая начать передачу в некоторый момент времени t, если обнаруживает электрические колебания в линии, то откладывает передачу до момента $t + t_d$, где t_d — задержка.

При работе сети каждая станция анализирует адресную часть передаваемых по сети кадров с це-

лью обнаружения и приема кадров, предназначенных для нее.

Конфликтом называют ситуацию, при которой две или более станции “одновременно” пытаются захватить линию. Понятие “одновременность событий” в связи с конечностью скорости распространения сигналов по линии конкретизируется как отстояние событий во времени не более чем на величину $2d$, называемую *окном столкновений*, где d — время прохождения сигналов по линии между конфликтующими станциями. Если какие-либо станции начали передачу в окне столкновений, то по сети распространяются искаженные данные. Это искажение и используют для обнаружения конфликта либо сравнением в передатчике данных, передаваемых в линию (неискаженных) и получаемых из нее (искаженных), либо по появлению постоянной составляющей напряжения в линии, что обусловлено искажением используемого для представления данных манчестерского кода. Обнаружив конфликт, станция должна оповестить об этом партнера по конфликту, послав дополнительный сигнал затора, после чего станции должны отложить попытки выхода в линию на время t_d. Очевидно, что значения t_d должны быть различными для станций, участвующих в столкновении (конфликте), поэтому t_d — случайная величина.

Маркерные методы доступа. Среди детерминированных методов преобладают *маркерные методы доступа*. Маркерный метод — метод доступа к среде передачи данных в ЛВС, основанный на передаче полномочий передающей станции с помощью специального информационного объекта, называемого маркером. Под полномочием понимается право инициировать определенные действия, динамически предоставляемые объекту, например станции данных в информационной сети.

Применяется ряд разновидностей маркерных методов доступа. Например, в *эстафетном методе* передача маркера выполняется в порядке очередности; в способе *селекторного опроса* (квантированной передачи) сервер опрашивает станции и передает полномочие одной из тех станций, которые готовы к передаче. В кольцевых одноранговых сетях широко применяют тактируемый маркерный доступ, при котором маркер циркулирует по кольцу и используется станциями для передачи своих данных.

2.4. Локальные вычислительные сети Ethernet

Состав аппаратуры. Одной из первых среди ЛВС шинной структуры была создана сеть Ethernet, разработанная фирмой Xerox. В этой сети был применен метод доступа МДКН/ОК. Позднее Ethernet стала основой стандарта IEEE 802/3. Другой вариант шинных ЛВС соответствует стандарту IEEE 802/4, описывающему сеть с эстафетной передачей маркера.

Технология Ethernet наиболее распространена в ЛВС. Так, по данным на 1996 г. 85% всех компьютеров в ЛВС были в сетях Ethernet.

В качестве линий передачи данных в ЛВС используют коаксиальный кабель, витую пару проводов или ВОЛС. Длины используемых отрезков коаксиального кабеля не должны превышать нескольких сотен метров, а у витой пары проводов — десятков метров. При больших расстояниях в среду передачи данных включают формирователи сигналов — повторители для сопряжения отрезков. ВОЛС позволяет существенно увеличить предельные расстояния и скорость передачи данных.

Для связи компьютеров со средой передачи данных используют сетевые контроллеры (адаптеры, сетевые карты), управляющие доступом к сети, и приемопередатчики, служащие для связи сетевого контроллера с линией связи.

Сетевой контроллер реализует принятый метод доступа к каналу, а также в случае метода МДКН/ОК осуществляет действия по выработке сигнала затора, задержке в передаче при наличии конфликта или при занятом моноканале, по формированию кадров, кодированию (декодированию) электрических сигналов в (из) специальный последовательный код, называемый манчестерским, распознаванию адреса в передаваемых по сети сообщениях.

После образования информационного кадра станция должна получить полномочия. Для этого контроллер прослушивает канал в ожидании его освобождения или прихода маркера. После получения полномочий осуществляется преобразование параллельного кода в последовательный, преобразование в манчестерский код и передача сигналов в кабель.

В состав приемопередатчика в шинных ЛВС с методом МДКН/ОК входят приемник сигналов от

линии и передатчик сигналов от станции в линию. Назначение приемника — усиление информационных сигналов и обнаружение конфликтов путем выделения постоянной составляющей искаженных сигналов и ее сопоставления в компараторе с эталонным напряжением.

Структура кадра. Кадр в стандарте IEEE 802/3, реализующем МДКН/ОК, имеет следующую структуру (ниже указаны последовательности полей кадра, их назначение, в скобках даны размеры полей в байтах):

< Преамбула (7) — ограничитель (1) — адрес назначения (2 или 6) — адрес источника (2 или 6) — длина кадра (2) — данные (от 64 до 1518 байт) — заполнение — контрольный код (4) >

Преамбула и ограничитель служат для установления синхронизации и отождествления начала кадра. Ограничители представляют собой уникальную последовательность битов, обычно это код 01111110. Чтобы эта последовательность была уникальной, в основных полях осуществляется *стаффинг* — добавление нуля после каждой последовательности из пяти подряд идущих единиц. На приемном конце такой нуль удаляется.

Шестибайтовый адрес — уникальный номер сетевой платы, он назначается изготовителем по выданной ему лицензии на определенный диапазон адресов.

Разновидности сетей Ethernet. Рядом фирм на базе проекта сети Ethernet разрабатывается оборудование для ЛВС. В настоящее время унифицировано несколько вариантов сети Ethernet, различающихся топологией, особенностями физической среды передачи данных, информационной скоростью передачи данных.

1. *Thick Ethernet (шина "с толстым" кабелем*); принятое обозначение варианта 10Base-5, где первый элемент "10" характеризует скорость передачи данных по линии 10 Мбит/с, последний элемент "5" — максимальную длину сегмента кабеля (в сотнях метров), т.е. 500 м. Другие параметры сети: максимальное число сегментов 5; максимальное число узлов на одном сегменте 100; минимальное расстояние между узлами 2,5 м. Здесь под *сегментом* кабеля понимается часть кабеля, используемая в качестве линии передачи данных и имеющая на концах согласующие элементы (*терминаторы)* для предотвращения отражения сигналов.

2. *Thin Ethernet (шина "с тонким" кабелем*, *cheapernet*); принятое обозначение 10Base-2: максимальное число сегментов 5; максимальная длина сегмента 185 м; максимальное число узлов на одном сегменте 30; минимальное расстояние между узлами 0,5 м; скорость передачи данных по линии 10 Мбит/с.

3. *Twisted Pair Ethernet;* принятое обозначение 10Base-T; это кабельная сеть с использованием витых пар проводов и концентраторов, называемых также распределителями, или хабами (hubs). Представление о структуре сети может дать рис. 2.4. В состав сетевого оборудования входят активные (AH) и пассивные (PH) концентраторы (active and passive hubs), различие между которыми заключается в наличии или отсутствии усиления сигналов и в количестве портов. Число портов в активных хабах обычно составляет 8, 12 или 16. В одной из разновидностей сети 10Base-T допускаются расстояния между активными распределителями до 600 м и между пассивными до 30 м, предельное число узлов 100. Физическая организация линий связи в 10Base-T мало напоминает шину. Однако в такой сети вполне возможна реализация метода доступа МДКН/ОК, и для пользователя (любого отдельного узла) разветвленная сеть из витых пар и концентраторов, по которой происходит широковещательная передача, есть просто среда передачи данных, такая же, как шина. Поэтому по логической организации сеть 10Base-T представляет собой сеть типа Ethernet. В то же время по своей топологии 10Base-T может быть вариантом "звезда", "дерево" и

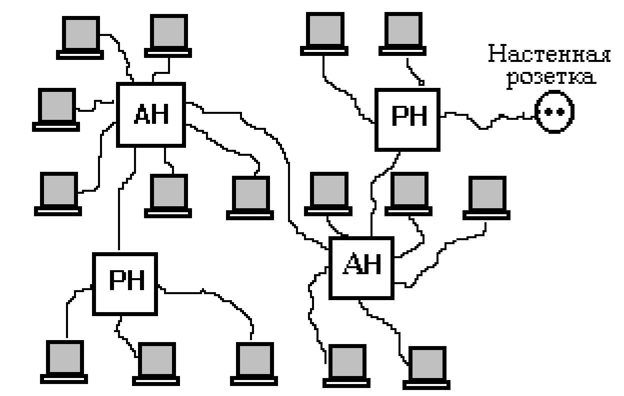


Рис. 2.4. Среда передачи данных на витой паре и концентраторах

т.п. В этой сети не рекомендуется включать последовательно более четырех хабов.

4. *Fiber Optic Ethernet* (шина на основе оптоволоконного кабеля), обозначение 10Base-F; применяется для соединений точка-точка, например, для соединения двух конкретных распределителей в кабельной сети. Максимальные длины — в пределах 2...4 км. Цена оптоволоконного кабеля приблизительно такая же, как и медного кабеля, но у первого из них меньше габариты и масса, достигается полная гальваническая развязка. Приемопередатчик (повторитель) для волоконно-оптических линий передачи данных (световодов) состоит из частей приемной, передающей, чтения и записи данных. В приемной части имеются фотодиод, усилитель-формирователь сигналов с требуемыми уровнями напряжения, механическое контактирующее устройство для надежного контакта фотодиода со стеклянной оболочкой кабеля. Передатчик представлен светодиодом или микролазером.

5. RadioEthernet (стандарт IEEE 802/11). Среда передачи данных — радиоволны, распространяющиеся в эфире. Структура сети может быть “постоянной” при наличии базовой кабельной сети с точками доступа от узлов по радиоканалам или “временной”, когда обмены между узлами происходят только по радиоканалам, Применяется модифицированный метод МДКН/ОК, в котором вместо обнаружения конфликтов используется предотвращение конфликтов. Это осуществляется следующим образом: узел, запрашивающий связь, посылает в эфир специальный кадр запроса, а передачу информации он может начать только после истечения межкадрового промежутка времени T, если за время T после запроса в эфире не было других запросов. Иначе попытка передачи откладывается на случайное время. Любой узел может посылать кадр запроса, только если за время T перед этим в эфире не было других кадров запроса.

Предусмотрена посылка положительной квитанции от приемного узла, подтверждающая правильность приема кадра. Квитанция посылается с малой задержкой t после окончания приема. В этом интервале длительностью t конфликты невозможны, так как претенденты на передачу могут посылать кадры запроса только в том случае, если перед посылкой эфир свободен в течение интервала времени не менее T (это условие выполняется и для узлов с отложенной из-за конфликта передачей), а $t<T$.

6. Сеть Fast Ethernet, иначе называемая 100BaseX или 100Base-T (стандарт IEEE 802/30). Информационная скорость 100 Мбит/с. В этой сети применен метод доступа МДКН/ОК. Используется для построения скоростных ЛВС (последовательно включается не более двух хабов), для объединения низкоскоростных подсетей 10Base-T в единую скоростную сеть и для подключения серверов на расстояниях до 200 м. В последнем случае серверы соединяются с клиентскими узлами через шину 100 Мбит/с и коммутатор, называемый также конвертором, преобразователем или переключателем скорости 100/10. К конвертору с другой стороны подключено несколько шин 10 Мбит/с, на которые нагружены остальные узлы. Практически можно использовать до 250 узлов, теоретически — до 1024. Подсетями могут быть как Fast Ethernet, так и обычные Ethernet со скоростью 10 Мбит/с, включенные через преобразователь скорости. Различают следующие варианты: 100Base-TX, в котором применяют кабель из двух неэкранированных витых пар категории 5, 100Base-T4 — с четырьмя неэкранированными парами категории 5, 100Base-FX — на ВОЛС.

7. Gigabit Ethernet 1000Base-X. В этом варианте получены гигабитные скорости. В соответствии со стандартом IEEE 802.3z имеются разновидности на ВОЛС с длиной волны 830 или 1270 нм (1000Base-SX и 1000Base-LX соответственно) на расстояниях до 550 м и на витой паре категории 5 (1000BaseCX) на расстояниях до 25 м. Скорость до 1 Гбит/с. Такая скорость достигается благодаря следующим решениям.

Сеть имеет иерархическую структуру. Участки (отдельные компьютеры или подсети) по 10 Мбит/с подключаются к портам переключателей (switches) скорости 10/100, их выходы по 100 Мбит/с, в свою очередь, подключаются к портам переключателей 100/1000. В сегментах сети, имеющих 1000 Мбит/с, используются, во-первых, передача данных по ВОЛС или параллельно по четырем витым парам, во-вторых, 5-уровневое представление данных (например, +2, +1, 0, -1, -2 В), в-третьих, кодирование 8b/10b (пояснение см. ниже). В результате в каждой витой паре имеем 250 Мбит/с при частоте сигналов 125 МГц, а это уже приемлемая частота для передачи по проводным соединениям.

2.5. Сети кольцевой топологии.

Сеть Token Ring. Из кольцевых ЛВС наиболее распространены сети с передачей маркера по кольцу и среди них: 1) ЛВС типа Token Ring (сеть с таким названием была разработана фирмой IBM и послужила основой для стандарта IEEE 802/5); 2) сети FDDI (Fiber Distributed Data Interface) на основе ВОЛС.

Кадр в стандарте IEEE 802/5 имеет структуру:

< Ограничитель (1) – управление доступом (1) – адрес назначения (6) – адрес источника (6) – данные (>=0) – контрольный код (4) – ограничитель (1) – состояние кадра (1) >

Поле “управление доступом” используется для указания порядкового номера кадра, смысла команд, содержащихся в кадре, и т.п. Так, в IEEE 802/5 это поле включает в себя указание приоритета (три бита), T — бит маркера, M — мониторный бит и три бита резервирования. Если $T = 0$, то кадр воспринимается как маркер, если $T = 1$, то кадр является информационным (т.е. маркер занят — поле “данные” заполнено). Поле “состояние кадра” используется для отметки того, что принимающая станция опознала свой адрес и восприняла данные.

Топология *сети Token Ring* показана на рис. 2.5,*а*. Концентраторы служат для удобства управления сетью, в частности для отключения от кольца неисправных узлов. На рис. 2.5,*б* представлена схема подключения узлов к кольцу в концентраторах. Для отключения узла достаточно левые переключатели (см. рис. 2.5,*б*) поставить в верхнее, а правые переключатели — в нижнее положение (в нормальном режиме положение переключателей противоположное).

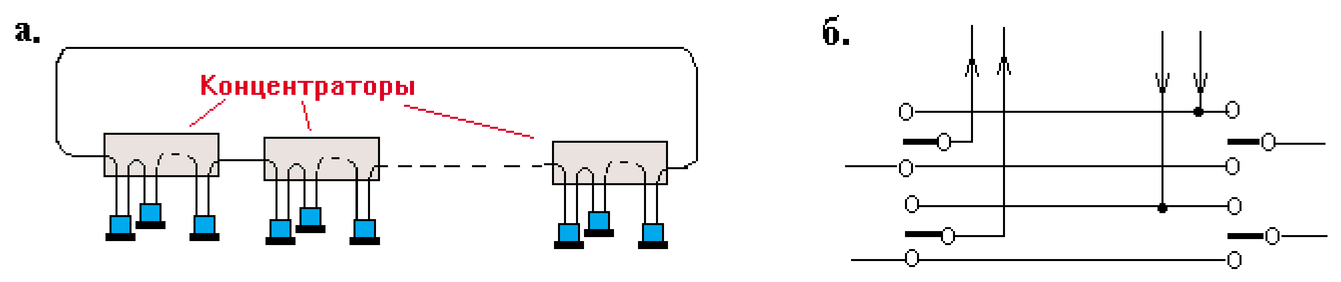


Рис. 2.5. Схема сети Token Ring: **а)** общий вид; **б)** схема подключения узла к кольцу

Типичная реализация сети Token Ring характеризуется следующими данными: максимальное число станций 96; максимальное число концентраторов 12; максимальная длина замыкающего кабеля 120 м; максимальная длина кабеля между двумя концентраторами или между концентратором и станцией 45 м; два варианта скорости передачи данных по линии 4 или 16 Мбит/с.

Функционирование сети заключается в следующем.

В кольцевых локальных сетях сигналы циркулируют по кольцу, состоящему из ряда отрезков линии связи, которые соединяют пары соседних узлов. Эти отрезки соединяются в узлах через повторители сигналов, выполняющих функции приема и передачи сигналов как из кольца и в кольцо, так и между АКД и линией. Повторители вносят некоторую задержку в передачу сигналов, поэтому общая задержка зависит от числа станций, включенных в кольцо.

Одним из способов взаимосвязи линии и АКД является способ вставки регистра. Станцию, получившую полномочия, называют *активной станцией*. Активная станция осуществляет вставку регистра в разрыв кольца и подключает передающий регистр, из которого в кольцо посылается передаваемый кадр.

Эти регистры являются сдвигающими. Кадр проходит через кольцо и возвращается на вставленный регистр. По пути его адресная часть проверяется остальными станциями, поскольку в них предусмотрена расшифровка адресной и управляющей информации. Если пакет предназначен данной станции, то принимается информационная часть пакета, проверяется правильность приема и при положительном результате проверки в кольцо направляется соответствующее подтверждение. Передающая

(активная) станция одновременно с передачей сформированного в ней пакета принимает пакет, прошедший по кольцу, на вставленный регистр. В каждом такте сдвига в кольцо направляется очередной бит данных, а из кольца с некоторой задержкой возвращаются переданные биты. Если подтверждена правильность передачи, то переданные данные стираются в передающей станции, которая направляет в кольцо свободный маркер, если не подтверждена, то осуществляется повторная передача пакета.

Станции, готовые к передаче собственных данных, ждут прихода свободного маркера. Станция, получившая полномочия, вставляет свой регистр в кольцо, становясь активной, а вставленный ранее регистр исключается из кольца.

Циркулирующий по сети маркер состоит из следующих частей:

<ограничитель - P - T - M - R - ограничитель>

Если $T = 0$, то маркер свободен. Тогда если он проходит мимо станции, имеющей данные для передачи, и приоритет станции не ниже значения, записанного в P, то станция преобразует маркер в информационный кадр: устанавливает $T = 1$ и записывает между R и конечным ограничителем адрес получателя, данные и другие сведения в соответствии с принятой структурой кадра. Информационный кадр проходит по кольцу, при этом происходит следующее: 1) каждая станция, готовая к передаче, записывает значение своего приоритета в R, если ее приоритет выше уже записанного в R значения; 2) станция-получатель, распознав свой адрес, считывает данные и отмечает в конце кадра (в бите “статус кадра”) факт приема данных.

Совершив полный оборот по кольцу, кадр приходит к станции-отправителю, которая анализирует состояние кадра. Если передача не произошла, то делается повторная попытка передачи. Если произошла, то кадр преобразуется в маркер указанной выше структуры с $T = 0$. При этом также происходят действия:

$$P := R; \; R := 0;$$

где P и R — трехбитовые коды.

При следующем обороте маркер будет захвачен той станцией-претендентом, у которой на предыдущем обороте оказался наивысший приоритет (именно его значение записано в P).

Сеть Token Ring рассчитана на меньшие предельные расстояния и число станций, чем сеть Ethernet, но лучше приспособлена к повышенным нагрузкам.

Сеть FDDI. *Сеть FDDI (Fiber Distributed Data Interface)* относится к высокоскоростным сетям, имеет кольцевую топологию, использует ВОЛС и специфический вариант маркерного метода доступа.

В основном варианте сети применено двойное кольцо на ВОЛС. Обеспечивается информационная скорость 100 Мбит/с. Расстояние между крайними узлами до 200 км, между соседними станциями — не более 2 км. Максимальное число узлов 500. В ВОЛС применяются волны длиной 1300 нм.

Два кольца ВОЛС используются одновременно. Станции можно подключать к одному из колец или к обоим сразу (рис. 2.6,*а*). Использование конкретным узлом обоих колец позволяет получить для этого узла суммарную пропускную способность 200 Мбит/с. Другое возможное использование второ-

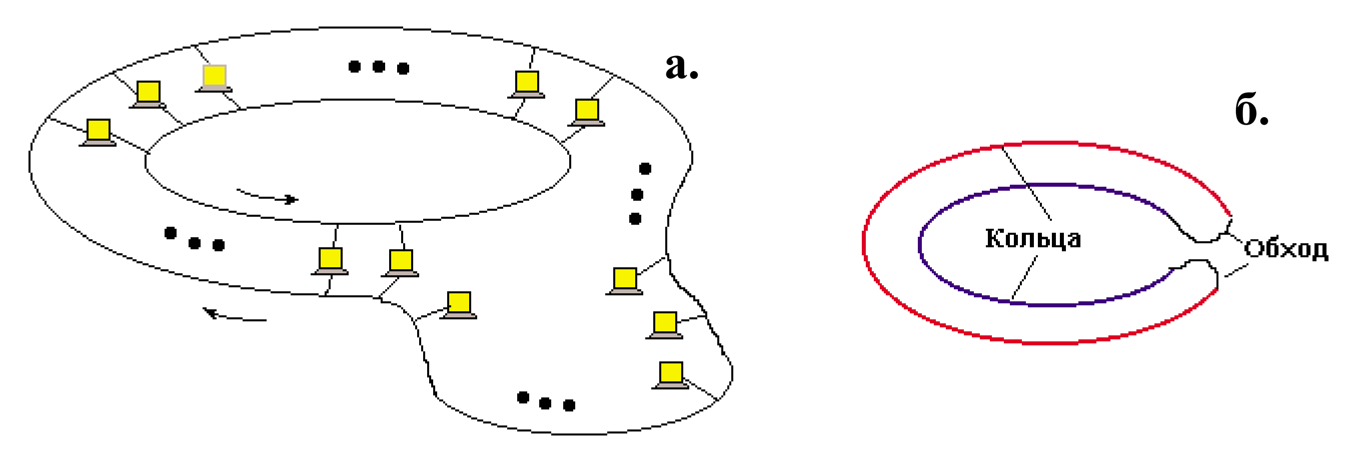


Рис. 2.6. Кольца ВОЛС в сети FDDI: **а)** включение узлов: **б)** обход поврежденного участка

го кольца — обход с его помощью поврежденного участка путем объединения колец, как показано на рис. 2.6,*б*.

В сети FDDI используются оригинальные код и метод доступа. Применяется код типа NRZ (без возвращения к нулю), в котором изменение полярности в очередном такте времени воспринимается как 1, отсутствие изменения полярности как 0. Чтобы код был самосинхронизирующимся, после каждых четырех битов передатчик вырабатывает синхронизирующий перепад.

Такое специальное манчестерское кодирование называют 4b/5b. Запись 4b/5b означает код, в котором для самосинхронизации при передаче четырех битов двоичного кода используется пять битов так, что не может быть более двух нулей подряд, или после четырех битов добавляется еще один обязательный перепад, что и используется в FDDI.

При использовании такого кода несколько усложняются блоки кодирования и декодирования, но при этом повышается скорость передачи по линии связи, так как почти вдвое уменьшается максимальная частота переключения по сравнению с манчестерским кодом.

В соответствии с методом FDDI по кольцу циркулирует пакет, состоящий из маркера и информационных кадров. Любая станция, готовая к передаче, распознав проходящий через нее пакет, вписывает свой кадр в конец пакета. Она же ликвидирует его после того, как кадр вернется к ней после полного оборота по кольцу и при условии, что он был воспринят получателем. Если обмен происходит без сбоев, то кадр, возвращающийся к станции-отправителю, оказывается в пакете уже первым, так как все предшествующие кадры должны быть ликвидированы раньше.

Сеть FDDI обычно используют как объединяющую в единую сеть многих отдельных подсетей ЛВС. Например, при организации информационной системы крупного предприятия целесообразно иметь ЛВС типа Ethernet или Token Ring в помещениях отдельных проектных подразделений, а связь между подразделениями осуществлять через сеть FDDI.

2.6. Каналы передачи данных в корпоративных сетях

Характеристики и типы каналов передачи данных. Применяемые в вычислительных сетях каналы передачи данных классифицируются по ряду признаков. Во-первых, по форме представления информации в виде электрических сигналов каналы подразделяют на цифровые и аналоговые. Во-вторых, по физической природе среды передачи данных различают каналы связи проводные (обычно медные), оптические (как правило, волоконно-оптические), беспроводные (инфракрасные и радиоканалы). В третьих, по способу разделения среды между сообщениями выделяют упомянутые выше каналы с временным (TDM) и частотным (FDM) разделением.

Одной из основных характеристик канала является его пропускная способность (скорость передачи информации, т.е. информационная скорость), определяемая полосой пропускания канала и способом кодирования данных в виде электрических сигналов. Информационная скорость измеряется количеством битов информации, переданных в единицу времени. Наряду с информационной оперируют *бодовой (модуляционной) скоростью,* которая измеряется в бодах, т.е. числом изменений дискретного сигнала в единицу времени. Именно бодовая скорость определяется полосой пропускания линии. Если одно изменение значения дискретного сигнала соответствует нескольким битам, то информационная скорость превышает бодовую. Действительно, если на бодовом интервале (между соседними изменениями сигнала) передается N бит, то число градаций сигнала равно 2^N. Например, при числе градаций 16 и скорости 1200 бод одному боду соответствует 4 бит/с и информационная скорость составляет 4800 бит/с. С ростом длины линии связи увеличивается затухание сигнала и, следовательно, уменьшаются полоса пропускания и информационная скорость.

Максимально возможная информационная скорость V связана с полосой пропускания F канала связи формулой Хартли-Шеннона (предполагается, что одно изменение значения сигнала приходится на $\log_2 k$ бит, где k — число возможных дискретных значений сигнала):

$$V = 2F \log_2 k \text{ бит/с},$$

так как $V = \log_2 k / t$, где t — длительность переходных процессов, приблизительно равная $3T_В$, а $T_В = 1 / (2\pi F)$. Здесь $k \le 1+A$, A — отношение сигнал/помеха.

Проводные линии связи в вычислительных сетях представлены коаксиальными кабелями и витыми парами проводов.

Используются коаксиальные кабели: “толстый” диаметром 12,5 мм и “тонкий” диаметром 6,25 мм. “Толстый” кабель имеет меньшее затухание, лучшую помехозащищенность, что обеспечивает возможность работы на больших расстояниях, но он плохо гнется, что затрудняет прокладку соединений в помещениях, и дороже “тонкого”.

Существуют экранированные (STP — Shielded Twist Pair) и неэкранированные (UTP — Unshielded Twist Pair) пары проводов. Экранированные пары сравнительно дороги. Чаще используются неэкранированные пары, имеющие несколько категорий (типов). Обычный телефонный кабель — пара категории 1. Пара категории 2 может использоваться в сетях с пропускной способностью до 4 Мбит/с. Витые пары имеют категории, начиная с третьей. Для сетей Ethernet (точнее, для ее варианта 10Base-T) разработана пара категории 3, а для сетей Token Ring — пара категории 4. Более совершенными являются неэкранированные витые пары категорий 5 и 6.

Пару категории 5 применяют при частотах до 100 МГц, в ней проводник представлен медными жилами диаметром 0,51 мм, навитыми по определенной технологии и заключенными в термостойкую изолирующую оболочку. В высокоскоростных ЛВС на UTP длины соединений обычно не превышают 100 м. Затухание в паре категории 5 на 100 МГц и при длине 100 м составляет около 24 дБ, при 10 МГЦ и 100 м — около 7 дБ.

Примерами пар категорий 6 и 7 могут служить кабели, выпускаемые фирмой PIC, в них размещается по 4 пары проводов, каждая со своим цветом полиэтиленовой изоляции. В случае категории 6 оболочка кабеля имеет диаметр 5 мм, используются медные проводники диаметром 0,5 мм, затухание на 100 МГц около 22 дБ. В случае категории 7 каждая пара дополнительно заключена в экранирующую алюминиевую фольгу, диаметр оболочки увеличен до 8 мм, затухание на 100 МГц составляет около 20 дБ, на 600 МГц — 50 дБ.

Витые пары иногда называют *сбалансированной* линией в том смысле, что в двух проводах линии передаются одни и те же уровни сигнала (по отношению к «земле»), но разной полярности. При приеме воспринимается разность сигналов, называемая парафазным сигналом. Синфазные помехи при этом самокомпенсируются.

Оптические линии связи реализуются в виде ВОЛС. ВОЛС являются основой высокоскоростной передачи данных, особенно на большие расстояния. В ЛВС каналы передачи данных представлены в основном проводными (медными) линиями, поскольку неэкранированные витые пары дешевле ВОЛС и удобнее при прокладке кабельной сети. Но для реализации высокоскоростных магистральных каналов в корпоративных и территориальных сетях ВОЛС уже находится вне конкуренции.

Конструкция ВОЛС — кварцевый сердечник диаметром 10 мкм, покрытый отражающей оболочкой с внешним диаметром 125...200 мкм. Типичные характеристики ВОЛС: работа на волнах 0,83...1,55 мкм, затухание 0,7 дБ/км, полоса частот — до 2 ГГц; ориентировочная цена – 4-5 долл. за 1 м. Предельные расстояния *D* для передачи данных по ВОЛС (без ретрансляции) зависят от длины волны излучения l: при l = 850 нм имеем *D* = 5 км, а при l = 300 нм имеем *D* = 50 км, но аппаратурная реализация дороже.

Примером среды передачи данных между мейнфреймами, рабочими станциями, пулами периферийных устройств может служить среда Fiber Channel на ВОЛС, обеспечивающая скорости от 133 до 1062 Мбит/с на расстояниях до 10 км (для сравнения приведем данные по стандартному интерфейсу SCSI между рабочей станцией и дисководом — скорость 160 Мбит/с при расстояниях не более десятков метров). На базе ВОЛС реализованы технологии передачи данных SDH (SONET) со скоростями 155 и 622 Мбит/с, рассматриваемые далее.

К числу новых стандартов для высокоскоростных магистралей передачи данных относятся стандарт цифровой синхронной иерархии SDH (Synchronous Digital Hierachy). В сетях SDH используют ВОЛС в качестве линий передачи данных. Стандарт устанавливает структуру фреймов, на которые разбивается поток передаваемых данных. Эта структура названа транспортным модулем.

Рассмотрим транспортный модуль STM-1. В нем фрейм состоит из 9-ти строк и 270 колонок, каждая позиция содержит 1 байт. В фрейме выделены три зоны. Первая зона содержит теги для разделения фреймов, для коммутации и управления потоком в промежуточных узлах (регенераторах оптических сигналов, устанавливаемых при больших длинах сегментов линии). Данные для управления в концевых узлах находятся во второй зоне. Третья зона содержит передаваемую информацию.

Информация конкретного сообщения может занимать ту или иную часть фрейма, называемую контейнером. Чем больше длина контейнера, тем выше информационная скорость. Предусмотрено

несколько типов контейнеров со скоростями 1,5; 6; 45 и 140 Мбит/с (по американскому стандарту) или 2; 6; 34 и 140 Мбит/с (по европейскому). Общая скорость передачи для STM-1 равна 155,52 Мбит/с.

Кроме STM-1, стандартом введены также модули STM-4 и STM-16 со скоростями 622 и 2488 Мбит/с соответственно.

Радиоканалы. В беспроводных радиоканалах передача информации осуществляется с помощью радиоволн. В информационных сетях используются диапазоны от сотен мегагерц до десятков гигагерц.

Для организации канала передачи данных в диапазонах дециметровых волн (902...928 МГц и 2,4...2,5 ГГц) требуется регистрация в Госсвязьнадзоре (1997 г.). Работа в диапазоне 5,725...5,85 ГГц пока лицензирования не требует.

Чем выше рабочая частота, тем больше емкость (число каналов) системы связи, но тем меньше предельные расстояния, на которых возможна прямая передача между двумя пунктами без ретрансляторов. Первая из причин и порождает тенденцию к освоению новых более высокочастотных диапазонов.

Радиоканалы используются в качестве альтернативы кабельным системам в локальных сетях и при объединении сетей отдельных подразделений и предприятий в корпоративные сети. Радиоканалы являются необходимой составной частью также в спутниковых и радиорелейных системах связи, применяемых в территориальных сетях, в сотовых системах мобильной связи.

Радиосвязь используют в корпоративных и локальных сетях, если затруднена прокладка других каналов связи. Во многих случаях построения корпоративных сетей применение радиоканалов оказывается более дешевым решением по сравнению с другими вариантами.

Радиоканал либо выполняет роль моста между подсетями (двухточечное соединение), либо является общей средой передачи данных в ЛВС по методу МДКН/ОК, либо служит соединением между центральным и терминальными узлами в сети с централизованным управлением, либо соединяет спутник с наземными станциями в спутниковом канале связи.

Радиомосты используют для объединения между собой кабельных сегментов и отдельных ЛВС в пределах прямой видимости и организации магистральных каналов в опорных сетях. Они выполняют ретрансляцию и фильтрацию пакетов. При этом имеет место двухточечное соединение с использованием направленных антенн, дальность в пределах прямой видимости (обычно до 15…20 км с расположением антенн на крышах зданий). Мост имеет два адаптера: один для формирования сигналов в радиоканале, другой – в кабельной подсети.

В случае использования радиоканала в качестве общей среды передачи данных в ЛВС сеть называют RadioEthernet (стандарт IEEE 802/11), обычно ее применяют внутри зданий. В состав аппаратуры входят приемопередатчики и антенны. Связь осуществляется на частотах от одного до нескольких гигагерц. Расстояния между узлами — несколько десятков метров.

В соответствии со стандартом IEEE 802/11 возможны два способа передачи двоичной информации в ЛВС, их цель заключается в обеспечении защиты информации от нежелательного доступа.

Первый способ называют *методом прямой последовательности* (DSSS — Direct Sequence Spread Spectrum). В нем защита информации основана на избыточности — каждый бит данных представлен последовательностью из 11-ти элементов (“чипов”). Эта последовательность создается с помощью алгоритма, известного участникам связи, и поэтому ее можно дешифрировать при приеме. Сохранение высокой скорости обеспечивается расширением полосы пропускания (в DSSS по IEEE 802/11 информационная скорость может доходить до 6 Мбит/с, полоса пропускания составляет 22 МГц в диапазоне частот 2,4 ГГц). Отметим, что избыточность повышает помехоустойчивость. Действительно, помехи обычно имеют более узкий спектр, чем 22 МГц, и могут исказить часть “чипов”, но высока вероятность того. что по остальным “чипам” значение бита будет восстановлено. При этом не нужно стремиться к большим значениям отношения сигнал/помеха, сигнал становится шумоподобным, что и обусловливает, во-первых, дополнительную защиту от перехвата, во-вторых, не создает помех, мешающих работе другой радиоаппаратуры.

Второй способ — *метод частотных скачков* (FHSS — Frequency Hopping Spread Spectrum). Согласно этому методу полоса пропускания по IEEE 802/11 делится на 79 поддиапазонов. Передатчик периодически (с шагом 20...400 мс) переключается на новый поддиапазон, причем алгоритм изменения частот известен только участникам связи и может изменяться, что и затрудняет несанкциониро-

ванный доступ к данным.

Вариант использования радиоканалов для связи центрального и периферийного узлов отличается тем, что центральный пункт имеет ненаправленную антенну, а в терминальных пунктах при этом применяются направленные антенны. Дальность связи составляет также десятки метров, а вне помещений — сотни метров.

Спутниковые каналы являются частью магистральных каналов передачи данных. В них спутники могут находиться на геостационарных (высота 36 тыс. км) или низких орбитах. В случае геостационарных орбит заметны задержки на прохождение сигналов (к спутнику и обратно около 500 мс). Возможно покрытие поверхности всего земного шара с помощью четырех спутников. В низкоорбитальных системах обслуживание конкретного пользователя происходит попеременно разными спутниками. Чем ниже орбита, тем меньше площадь покрытия и, следовательно, требуется или больше наземных станций, или необходима межспутниковая связь, что, естественно, приводит к утяжелению спутника. Число спутников также значительно больше (обычно несколько десятков)

Поставкой оборудования для организации корпоративных и локальных беспроводных сетей занимается ряд фирм, в том числе известные фирмы Lucent Technologies, Aironet, Multipoint Network.

В оборудование беспроводных каналов передачи данных входят сетевые адаптеры и радиомодемы, поставляемые вместе с комнатными антеннами и драйверами. Они различаются способами обработки сигналов, характеризуются частотой передачи, пропускной способностью, дальностью связи.

Сетевой адаптер вставляют в свободный разъем шины компьютера. Например, адаптер WaveLAN (Lucent Technologies) подключают к шине ISA, он работает на частоте 915 МГц, пропускная способность 2 Мбит/с.

Радиомодем подключают к цифровому ООД через стандартный интерфейс.

Например, радиомодемы серии RAN (Multipoint Networks) могут работать в дуплексном или полудуплексном режиме; со стороны порта данных — интерфейс RS-232C, RS-449 или V.35, скорости до 128 кбит/с; со стороны радиопорта — частоты 400...512 или 820...960 МГц, ширина радиоканала 25...200 кГц.

В вычислительных сетях САПР в основном используются *цифровые каналы передачи данных*. Однако применяют и *аналоговые каналы*, поскольку таковыми являются телефонные сети, которые можно использовать в качестве магистральных каналов или абонентских линий.

Аналоговые каналы. В телефонных каналах общего пользования полоса пропускания составляет 0,3...3,4 кГц (каналы с такой полосой пропускания называют каналами тональной частоты), что соответствует спектру человеческой речи.

Для передачи дискретной информации по каналам тональной частоты необходимы устройства преобразования сигналов, согласующие характеристики дискретных сигналов и аналоговых линий. Такое преобразование называют модуляцией при передаче и демодуляцией при приеме и осуществляют с помощью специальных устройств — *модемов*.

Модуляция осуществляется с помощью воплощения сигнала, выражающего передаваемое сообщение, в некотором процессе, называемом переносчиком и приспособленном к реализации в данной среде. Переносчик в системах связи представляет собой электромагнитные колебания U некоторой частоты, называемой несущей частотой:

$$U = U_m \sin(\varpi t+\psi),$$

где U_m — амплитуда, ϖ — частота, ψ — фаза колебаний несущей. Если сообщение переносится на амплитуду U_m, то модуляцию называют *амплитудной* (АМ), если на частоту ϖ — *частотной* (ЧМ), и если на фазу ψ — *фазовой* (ФМ).

Для повышения информационной скорости применяют квадратурно-амплитудную модуляцию (QAM — Quadrature Amplitude Modulation, ее также называют квадратурно-импульсной), которая основана на передаче одним элементом модулированного сигнала n бит информации, где n = 4...8 (т.е. используются 16...256 дискретных значений амплитуды). Однако, чтобы правильно различать эти значения амплитуды, требуется малый уровень помех (отношение сигнал/помеха не менее 12 дБ при n = 4). При меньших отношениях сигнал/помеха лучше применять фазовую модуляцию с четырьмя или восемью дискретными значениями фазы для представления 2 или 3 битов информации соответственно.

Современные высокоскоростные модемы построены в соответствии с протоколами V.32 или V.34. В протоколе V.34 скорости составляют от 2,4 до 28,8 кбит/с с шагом 2,4 кбит/с. Протокол преду-

сматривает адаптацию передачи под конкретную обстановку при изменении несущей в пределах 1600...2000 Гц, а также автоматическое предварительное согласование способов модуляции в вызывающем и вызывном модемах. В протоколе V34.bis скорости могут достигать 33,6 кбит/с. В последнее время стали выпускаться модемы на 56 кбит/с по технологиям, названным x2 и V.90.

Цифровые каналы. Для передачи аналоговых сигналов по цифровым каналам связи применяют импульсно-кодовую модуляцию (ИКМ или PCM — Pulse Code Modulation). Этот вид модуляции сводится к измерению амплитуды аналогового сигнала в моменты времени, отстоящие друг от друга на *dt*, и к кодированию этих амплитуд цифровым кодом. Согласно *теореме Котельникова* величину *dt* определяют следующим образом: для неискаженной передачи должно быть не менее двух отсчетов на период колебаний, соответствующий высшей составляющей в частотном спектре сигнала. Требуемую пропускную способность определяют, исходя из условия обеспечения передачи голоса с частотным диапазоном до 4 кГц при кодировании восемью (или семью) битами. Отсюда получаем, что частота отсчетов (передачи байтов) равна 8 кГц, т.е. биты передаются с частотой 64 кГц (или 56 кГц при семибитовой кодировке).

В *цифровых каналах* для представления двоичной информации преимущественно используют самосинхронизирующийся *манчестерский код*. Пример манчестерского кода представлен на рис. 2.7. Самосинхронизация избавляет от необходимости иметь дополнительную линию связи для передачи синхронизирующих импульсов. Самосинхронизация обеспечивается благодаря формированию синхроимпульсов из перепадов, имеющихся в каждом такте манчестерского кода.

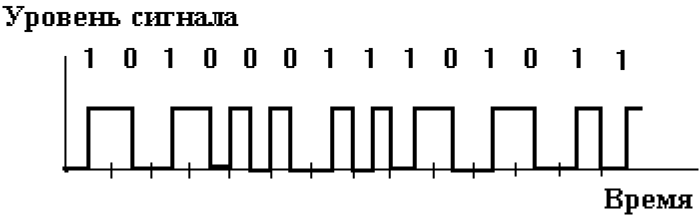


Рис. 2.7. Манчестерское кодирование

Различают несколько технологий связи, основанных на цифровых каналах.

В качестве магистральных каналов передачи данных в США и Японии применяют стандартную *многоканальную систему* T1 (иначе DS-1). Она включает в себя 24 цифровых канала, называемых DS-0 (Digital Signal-0). В каждом канале применена импульсно-кодовая модуляция с частотой следования отсчетов 8 кГц и квантованием сигналов по $2^8 = 256$ уровням, что обеспечивает скорость передачи 64 кбит/с на один канал или 1554 кбит/с на аппаратуру T1. В Европе широко распространена аппаратура E1 с 32 каналами по 64 кбит/с, т.е. с общей скоростью 2048 кбит/с. Применяются также каналы T3 (или DS-3), состоящие из 28 каналов T1 (45 Мбит/с) и E3 (34 Мбит/с) преимущественно в частных высокоскоростных сетях.

В канале T1 использовано временное мультиплексирование (TDM). Все 24 канала передают в мультиплексор по одному байту, образуя 192-битный кадр с добавлением одного бита синхронизации. Суперкадр составляют 24 кадра. В нем имеются контрольный код и синхронизирующая комбинация. Сборку информации из нескольких линий и ее размещение в магистрали T1 осуществляет мультиплексор. Канал DS-0 (один слот) соответствует одной из входных линий, т.е. реализуется коммутация каналов. Некоторые мультиплексоры позволяют маршрутизировать потоки данных, направляя их в другие мультиплексоры, связанные с другими каналами T1, хотя собственно каналы T1 называют некоммутируемыми.

При обычном мультиплексировании каждому соединению выделяется определенный слот (например, канал DS-0). Если же этот слот не используется из-за недогрузки канала по этому соединению, но по другим соединениям трафик значительный, то эффективность будет невысокой. Загружать свободные слоты или, другими словами, динамически перераспределять слоты можно, используя так называемые *статистические мультиплексоры* на основе микропроцессоров. В этом случае временно весь канал DS-1 или его часть отдается одному соединению с указанием адреса назначения.

В современных сетях важное значение имеет передача как данных, представляемых дискретными сигналами, так и аналоговой информации (например, голос и видеоизображения первоначально имеют аналоговую форму). Поэтому для многих применений современные сети должны быть *сетями интегрального обслуживания*. Наиболее перспективными сетями интегрального обслуживания являются сети с цифровыми каналами передачи данных, например, сети ISDN.

Сети ISDN могут быть коммутируемыми и некоммутируемыми. Различают обычные ISDN со скоростями от 56 кбит/с до 1,54 Мбит/с и широкополосные ISDN (Broadband ISDN, или B-ISDN) со скоростями 155... 2048 Мбит/с.

Применяют два варианта обычных сетей ISDN — базовый и специальный. В базовом варианте имеются два канала по 64 кбит/с (эти каналы называют *B*-каналами) и один служебный канал с 16 кбит/с (*D*- канал). В специальном варианте — 23 канала *B* по 64 кбит/с и один или два служебных канала *D* по 16 кбит/с. Каналы *B* можно использовать как для передачи закодированной голосовой информации (коммутация каналов), так и для передачи пакетов. Служебные каналы используются для сигнализации — передачи команд, в частности, для вызова соединения.

Очевидно, что для реализации технологий T1, T3, ISDN необходимо выбирать среду передачи данных с соответствующей полосой пропускания.

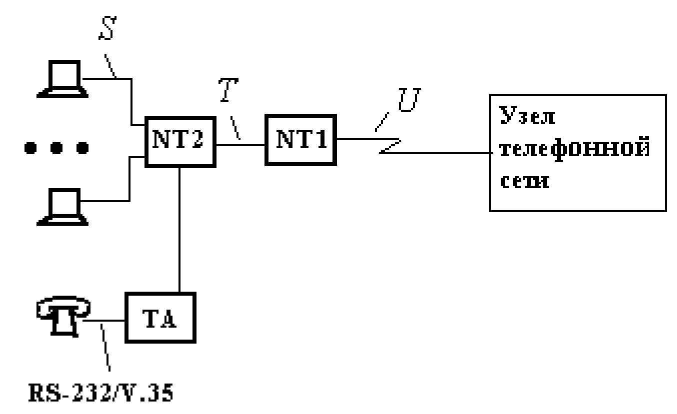


Рис. 2.8. Схема ISDN

Схема ISDN показана на рис. 2.8. Здесь *S*-соединение представляет собой четырехпроводную витую пару. Если оконечное оборудование не имеет интерфейса ISDN, то его подключают к *S* через специальный адаптер TA. Устройство NT2 объединяет *S*-линии в одну *T*-шину, которая имеет два провода от передатчика и два — к приемнику. Устройство NT1 реализует схему эхо-компенсации (рис. 2.9) и служит для интерфейса *T*-шины с обычной телефонной двухпроводной абонентской линией *U*.

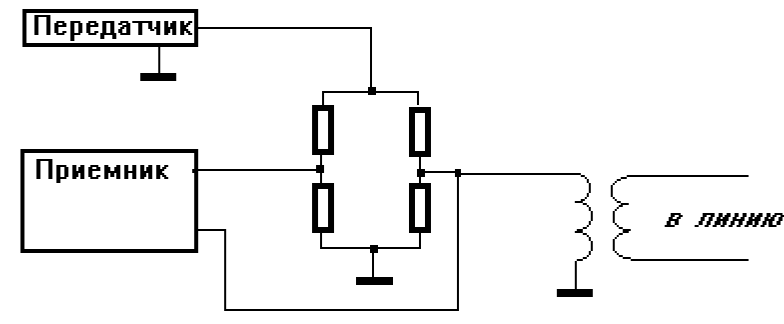


Рис. 2.9. Эхо-компенсация

Для подключения клиентов к узлам магистральной сети с применением на “последней миле” обычного телефонного кабеля наряду с каналами ISDN можно использовать *цифровые абонентские линии* xDSL. К их числу относятся HDSL (High-bit-rate Digital Subcriber Loop), SDSL (Single Pair Symmetrical Digital Subcriber Loop), ADSL (Asymmetric Digital Subcriber Loop). Например, в HDSL используют две пары проводов, амплитудно-фазовая модуляция без несущей, пропускная способность до 2 Мбит/с, расстояния до 7,5 км. Применяемые для кодирования устройства также называют модемами. Собственно ISDN можно рассматривать, как разновидность xDSL.

Организация дуплексной связи. Для организации дуплексной связи, т.е. одновременной передачи информации по линии в обоих направлениях используют следующие способы:
четырехпроводная линия связи — одна пара проводов для прямой и другая — для обратной передачи, что, естественно, дорого;

ч*астотное разделение* — прямая и обратная передачи ведутся на разных частотах, но при этом полоса для каждого направления сужается более чем вдвое по сравнению с полосой симплексной (однонаправленной) связи;

эхо-компенсация — при установлении соединения с помощью посылки зондирующего сигнала определяются параметры (запаздывание и мощность) эха — отраженного собственного сигнала; в дальнейшем из принимаемого сигнала вычитается эхо собственного сигнала (см. рис.2.9).

2.7. Стеки протоколов и типы сетей в автоматизированных системах

Протокол TCP. Протоколы, используемые совместно в сетях определенного типа, объединяются в совокупности, называемые *стеками протоколов*. Широко известны стеки протоколов TCP/IP, SPX/IPX, X.25, Frame Relay (FR), ATM, семиуровневые протоколы ЭМВОС.

Наибольшее распространение получили протоколы TCP/IP в связи с их использованием в каче-

стве основных в сети Internet. TCP/IP — пятиуровневые протоколы, но базовыми среди них, давшими название всей совокупности, являются протокол транспортного уровня TCP (Transmission Control Protocol) и протокол сетевого уровня IP (Internet Protocol). Эти протоколы поддерживаются такими операционными системами, как Unix и Windows-95/NT.

TCP — *дуплексный транспортный протокол* с установлением соединения. Под установлением соединения подразумевают установление виртуального канала в сети путем обмена запросом и согласием на соединение между отправителем и получателем сообщения. К другим функциям TCP относятся упаковка и распаковка пакетов на концах транспортного соединения; управление потоком — получатель одновременно с подтверждением правильности передачи сообщает размер окна, т.е. число пакетов, которые получатель готов принять или, что практически то же самое, число пакетов, которые отправитель может послать в сеть, не дожидаясь получения подтверждения об их правильном приеме; помещение срочных данных между специальными указателями, т.е. возможность управлять скоростью передачи.

В программном обеспечении протокола TCP имеется программа-агент, которая постоянно готова к работе и при приходе запроса генерирует свою копию для обслуживания создаваемого соединения, а сама программа-родитель ждет новых вызовов.

В схеме установления соединения в сетях клиент-сервер предусмотрена посылка клиентом запроса на соединение (команда ACTIVE_OPEN) с указанием адреса сервера, тайм-аута (времени жизни), уровня секретности. Можно сразу же поместить в запрос данные (тогда используется команда ACTIVE_OPEN_WITH_DATA). Если сервер готов к связи, он отвечает командой согласия (OPEN_RECEIVED), в которой назначает номер соединения. Далее командой SEND посылаются данные, а командой DELIVER подтверждается их получение. Разъединение выполняется обменом командами CLOSE и CLOSING.

Структура TCP-пакета (в скобках указано число битов):

порт отправителя (16)
порт получателя (16)
код позиции в сообщении, т.е. порядковый номер первого байта в поле данных (32)
номер следующего байта (32)
управление (16)
размер окна, т.е. число байт, которое можно послать до получения подтверждения (16)
контрольный код (16)
дополнительные признаки, например срочность передачи (16)
опции (24)
заполнитель (8)
данные

Следует отметить, что каждый байт сообщения получает уникальный порядковый номер. Отсюда вытекает одно из ограничений на максимально допустимую в протоколе TCP/IP пропускную способность. Это ограничение составляет 2^{32} байта / время жизни дейтаграммы, так как для конкретного соединения в сети не должно одновременно существовать более одного байта с одним и тем же номером.

Еще более жесткое ограничение возникает вследствие представления размера окна 16-ю битами. Это ограничение заключается в том, что за время T_v прохождения пакета от отправителя к получателю и обратно в сеть может быть направлено не более 2^{16} информационных единиц конкретного сообщения. Поскольку обычно такой единицей является байт, то имеем (2^{16} * 8 бит) / T_v. Так, для каналов со спутниками на геостационарных орбитах T_v составляет около 0,5 с и ограничение скорости будет около 1 Мбит/с. Можно заметно увеличить этот предел, если в качестве информационной единицы использовать C байт, $C > 1$.

В протоколе TCP повторная передача пакета происходит, если в течение оговоренного интервала времени T_m (тайм-аута) от получателя не пришло положительное подтверждение правильного приема. Обычно T_m=2 t , где t — некоторая оценка времени T_v прохождения пакета в обе стороны. Это время периодически корректируется по результату измерения T_v, а именно

$$t := 0{,}9\, t + 0{,}1 T_v.$$

Попытки повторных передач пакета не могут продолжаться бесконечно, и при превышении интервала времени, устанавливаемого в пределах 0,5...2,0 мин, соединение разрывается.

Размер окна регулируют следующим образом. Если сразу же после установления соединения выбрать завышенный размер окна, что означает разрешение посылки пакетов с высокой интенсивностью, то велика вероятность появления перегрузки определенных участков сети. Поэтому используют алгоритм так называемого медленного старта. Сначала посылается один пакет и после подтверждения его приема окно увеличивается на единицу, т.е. посылается два пакета. Если вновь приходит положительное подтверждение (потерь пакетов нет), то посылается уже четыре пакета и т.д. Скорость растет, пока пакеты проходят успешно. При потере пакета или при приходе от протокола управления сигнала о перегрузке размер окна уменьшается и далее возобновляется процедура линейного роста размера окна. Медленный старт снижает информационную скорость, особенно при пересылке коротких сообщений, поэтому стараются применять те или иные приемы его улучшения.

Протокол IP. *Сетевой протокол* IP — дейтаграммный сетевой протокол, т.е. протокол без установления соединения.

В *дейтаграммных протоколах* сообщение разбивается на дейтаграммы. Дейтаграмма — это пакет, передаваемый независимо от других частей одного и того же сообщения в вычислительных сетях с коммутацией пакетов. Дейтаграммы одного и того же сообщения могут передаваться в сети по разным маршрутам и поступать к адресату в произвольной последовательности, что требует дополнительных операций по сборке сообщения из дейтаграмм в узле-получателе. На внутренних участках маршрута контроль правильности передачи не предусмотрен и надежность связи обеспечивается лишь контролем в оконечном узле.

К функциям протокола IP относятся фрагментация и сборка пакетов при прохождении через промежуточные сети, имеющие другие протоколы; маршрутизация, т.е. определение пути прохождения пакета по разветвленной сети; проверка контрольного кода заголовка пакета (правильность передачи всего пакета проверяется на транспортном уровне, т.е. с помощью TCP, в оконечном узле); управление потоком — сброс дейтаграмм при превышении заданного времени жизни.

Структура дейтаграммы в IP (в скобках указано число битов):

версия протокола IP (4)
длина заголовка (4)
тип сервиса (8)
общая длина (16)
идентификация — порядковый номер дейтаграммы (16)
место фрагмента в дейтаграмме, т.е. номер фрагмента, используемый при фрагментации дейтаграммы в промежуточных сетях (16)
время жизни дейтаграммы в сети (8)
тип протокола (8)
контрольный код CRC заголовка (16)
адрес источника (32)
адрес назначения (32)
опции (32)
данные (не более 65356 байт)

От версии протокола зависит структура заголовка. Сделано это для возможности последующего внесения изменений. Например, предполагается вместо четырехбайтовых адресов установить в

Internet в будущем шестибайтовые адреса.

В поле “Тип сервиса” задается приоритет (если приоритетность используется), можно указать одно из следующих требований: минимальная задержка, высокая надежность, низкая цена передачи данных.

Всего в сети одновременно может быть 2^{16} = 65 тысяч дейтаграмм сообщения с разными идентификаторами, т.е. за отрезок времени, равный времени жизни дейтаграммы, может быть передано не более 2^{16} дейтаграмм. Это один из факторов, ограничивающих пропускную способность сетей с протоколом IP. Действительно, при времени жизни 120 с имеем предельную скорость $2^{16}/ 120$ = 546 дейтаграмм в секунду, что при размере дейтаграммы до 65 тысяч байт дает ограничение скорости приблизительно в 300 Мбит/с (такое же значение одного из ограничений предельной скорости получено выше и для протокола TCP).

Время жизни может измеряться как в единицах времени T , так и в хопах P (числом пройденных маршрутизаторов). В первом случае контроль ведется по записанному в заголовке значению T, которое уменьшается на единицу каждую секунду. Во втором случае каждый маршрутизатор уменьшает число P, записанное в поле “Время жизни”, на единицу. При $T = 0$ или при $P = 0$ дейтаграмма сбрасывается.

Поле “Тип протокола” определяет структуру данных в дейтаграмме. Примерами протоколов могут служить UDP, SNA, IGP и т.п.

Поле “Опции” в настоящее время рассматривается как резервное.

Адресация в TCP/IP. В TCP/IP различают два типа адресов. На канальном уровне используют адреса, называемые *физическими*. Это шестибайтовые адреса сетевых плат, присваиваемые изготовителем контроллеров (как уже отмечалось, каждый изготовитель вместе с лицензией на изготовление получает уникальный диапазон адресов). На сетевом уровне используют сетевые адреса, иначе называемые *виртуальными,* или *логическими*.

Различают понятия сетевых адреса и имени, имеющих цифровое и буквенное выражения соответственно.

Сетевой адрес называют IP-адресом. Это четырехбайтовый код, состоящий из двух частей: адреса сети и адреса узла (заметим, что узел, имеющий IP-адрес, называют хостом). Имя характеризует пользователя. Его составляют в соответствии с доменной системой имен. Соответствие между IP-адресом и IP-именем хоста устанавливается специальной *службой имен*. В Internet это DNS (Domain Name Service), в семиуровневой модели ISO — стандарт X.500.

IP-имя, называемое также *доменным именем*, — удобное для человека название узла или сети. Имя отражает иерархическое построение глобальных сетей и потому состоит из нескольких частей (аналогично обычным почтовым адресам). Корень иерархии обозначает либо страну, либо отрасль знаний, например: ru — Россия, us — США, de — Германия, uk — Великобритания, edu — наука и образование, com — коммерческие организации, org — некоммерческие организации, gov — правительственные организации, mil — военные ведомства, net — служба поддержки Internet и т.д. Корень занимает в IP-имени правую позицию, левее записываются локальные части адреса и, наконец, перед символом @ указывается имя почтового ящика пользователя. Так, запись norenkov@wwwcdl.bmstu.ru расшифровывается следующим образом: пользователь norenkov имеет почтовый ящик в сервере wwwcdl организации bmstu в стране ru. Уже к 1998 г. число используемых доменных имен в сети Internet превысило один миллион.

IP-адрес — 32-битовое слово, записываемое в виде четырех частей (побайтно), разделенных точками. Каждые подсеть и узел в подсети получают свои номера, причем для сети (подсети) можно использовать от одного до трех старших байтов, а оставшиеся байты — для номера узла. Какая часть IP-адреса относится к сети, определяется ее маской, выделяющей соответствующие биты в IP-адресе. Например, для некоторой сети маска может быть 255.0.0.0, а для ее подсети — 255.255.0.0 и т.д. Тем самым описывается иерархия сетей.

Номера при включении нового хоста выдает организация-провайдер, предоставляющая телекоммуникационные услуги. Провайдер, в частности, обеспечивает включение IP-адреса и соответствующего ему IP-имени в сервер службы адресов DNS. Это означает запись данных о хосте в DIB (Dire-

ctory Information Base) локального узла DNS.

При маршрутизации пользователь, отправляющий сообщение, задает IP-имя получателя. Поскольку маршрутизация в сети осуществляется по IP-адресам, то с помощью серверов DNS осуществляется перевод указанного IP-имени в IP-адрес.

В локальной сети, где используются шестибайтовые адреса, называемые MAC-адресами, требуется преобразование IP-имен в MAC-адреса. Это преобразование выполняет маршрутизатор, связывающий локальную сеть с территориальной, в соответствии с специальным протоколом ARP, имеющимся в стеке TCP/IP. Для этого в маршрутизаторе создается таблица соответствия IP-имен и MAC-адресов данной сети.

Маршрутизация в Internet организована по иерархическому принципу. Имеются уровни ЛВС и корпоративных сетей; маршрутных доменов, в каждом из которых используются единые протоколы и алгоритмы маршрутизации; административных доменов, каждый из которых соответствует некоторой ассоциации и имеет единое управляющее начало. В маршрутных доменах имеются внешние маршрутизаторы для связи с другими маршрутными или административными доменами. Обращение из некоторого узла к Internet (например, из wwwcdl.bmstu.ru по адресу http://www.eevl.ac.uk) происходит к местному серверу (bmstu), и если там сведений об адресе назначения нет, то происходит переход к серверу следующего, более высокого уровня (ru) и далее по иерархии вниз до получения IP-адреса хоста назначения. В местном DNS-сервере могут быть сведения об IP-адресах хостов из удаленных доменов, если к ним происходят достаточно частые обращения из данного домена.

Другие протоколы стека TCP/IP. В стек протоколов TCP/IP входит ряд других протоколов. Например, на транспортном уровне это *протокол UDP* (User Datagram Protocol) — транспортный протокол без установления соединения, он значительно проще TCP, но его используют чаще всего для сообщений, умещающихся в один пакет. После оформления UDP-пакета он передается с помощью средств IP к адресату, который по заголовку IP-пакета определяет тип протокола и передает пакет не агенту TCP, а агенту UDP. Агент определяет номер порта и ставит пакет в очередь к этому порту. В UDP служебная часть дейтаграммы короче, чем в TCP (8 байт вместо 20), не требуется предварительного установления соединения или подтверждения правильности передачи, как это делается в TCP, что и обеспечивает большую скорость за счет снижения надежности доставки.

Структура UDP-дейтаграммы (в скобках указано число битов):

```
порт отправителя (16)
порт получателя (16)
длина (16)
контрольная сумма (16)
данные (не более 65356 байт)
```

Протоколы более высоких уровней, чем TCP, в сетях TCP/IP называют *прикладными* протоколами. В частности, к ним относят следующие протоколы:

— SMTP (Simple Mail Transport Protocol) — почтовый протокол, который по классификации ЭМВОС наиболее близок к прикладному уровню;

— FTP (File Transfer Protocol) — протокол с функциями представительного по ЭМВОС уровня;

— Telnet — протокол с функциями сеансового по ЭМВОС уровня.

На нижних уровнях в TCP/IP используются протоколы IEEE 802/X или X.25.

Для управления сетью в стек TCP/IP включены специальные *протоколы управления*.

Среди протоколов управления различают протоколы, реализующие управляющие функции сетевого уровня, и протоколы мониторинга за состоянием сети, относящиеся к более высоким уровням. В сетях TCP/IP роль первых из них выполняет протокол ICMP (Internet Control Message Protocol), роль вторых — протокол SNMP (Simple Network Management Protocol).

Основные функции *ICMP*:

— оповещение отправителя с чрезмерным трафиком о необходимости уменьшить интенсивность посылки пакетов; при перегрузке адресат (или промежуточный узел) посылает ICMP-пакеты,

указывающие о необходимости сокращения интенсивности входных потоков;

— передача откликов (квитанций) на успешно переданные пакеты;

— контроль времени жизни T дейтаграмм и их ликвидация при превышении T или по причине искажения данных в заголовке;

— оповещение отправителя о недостижимости адресата; отправление ICMP-пакета с сообщением о невозможности достичь адресата осуществляет маршрутизатор;

— формирование и посылка временных меток (измерение задержки) для контроля T_v — времени доставки пакетов, что нужно для "оконного" управления. Например, время доставки T_v определяется следующим образом. Отправитель формирует ICMP-запрос с временной меткой и отсылает пакет. Получатель меняет адреса местами и отправляет пакет обратно. Отправитель сравнивает метку с текущим временем и тем самым определяет T_v .

ICMP-пакеты вкладываются в IP-дейтаграммы при доставке.

Основные функции протоколов мониторинга заключаются в сборе информации о состоянии сети, предоставлении этой информации нужным лицам путем посылки ее на соответствующие узлы, возможном автоматическом принятии необходимых управляющих мер.

Собственно собираемая информация о состоянии сети хранится в базе данных под названием MIB (Management Information Base). Примеры данных в MIB: статистика по числу пакетов и байтов, отправленных или полученных правильно или с ошибками, длины очередей, максимальное число соединений и др.

Протокол SNMP относится к прикладному уровню в стеке протоколов TCP/IP. Он работает по системе менеджер-агент. Менеджер (серверная программа SNMP) посылает запросы агентам, агенты (т.е. программы SNMP объектов управления) устанавливаются в контролируемых узлах, они собирают информацию (например, о загрузке, очередях, временах совершения событий) и передают ее серверу для принятия нужных мер. В общем случае агентам можно поручить и обработку событий, и автоматическое реагирование на них. Для этого в агентах имеются триггеры, фиксирующие наступление событий, и средства их обработки. Команды SNMP могут запрашивать значения объектов MIB, посылать ответы, менять значения параметров.

Чтобы послать команду SNMP, используют транспортный протокол UDP.

Одной из проблем управления по SNMP является защита агентов и менеджеров от ложных команд и ответов, которые могут дезорганизовать работу сети. Используется шифрование сообщений, но это снижает скорость реакции сети на происходящие события.

Расширением SNMP являются протоколы RMON (Remote Monitoring) для сетей Ethernet и Token Ring и RMON2 для сетевого уровня. Преимущество RMON заключается в меньшем трафике, так как здесь агенты более самостоятельны и сами выполняют часть необходимых управляющих воздействий на состояние контролируемых ими узлов.

На базе протокола SNMP разработан ряд мощных средств управления, примерами которых могут служить продукт ManageWISE фирмы Novell или система UnicenterTNG фирмы Computer Associates. С их помощью администратор сети может выполнять следующие действия: 1) строить 2*D* изображение топологии сети, причем на разных иерархических уровнях, перемещаясь от региональных масштабов до подсетей ЛВС (при интерактивной работе); 2) разделять сеть на домены управления по функциональным, географическим или другим принципам с установлением своей политики управления в каждом домене; 3) разрабатывать нестандартные агенты с помощью имеющихся инструментальных средств.

Дальнейшее развитие подобных систем может идти в направлении связи сетевых ресурсов с проектными или бизнес-процедурами и сетевых событий с событиями в процессе проектирования или управления предприятиями. Тогда система управления сетью станет комплексной системой управления процессами проектирования и управления предприятием.

Рассмотрим другие стеки протоколов.

Протоколы SPX/IPX. В сетях Netware фирмы Novell используются протоколы SPX *(Sequence Packet Exchange)* и IPX *(Internet Packet Exchange)* для транспортного и сетевого уровней соответственно.

Адрес получателя в пакете IPX состоит из номера сети (фактически номера сервера), адреса узла (это имя сетевого адаптера) и имени гнезда (прикладной программы). Пакет имеет заголовок в 30 байт и блок данных длиной до 546 байт. В пакете SPX заголовок включает 42 байт, т.е. блок данных

не более 534 байт.

Установление виртуального соединения в SPX (создание сессии) заключается в посылке клиентом запроса connect, возможная реакция сервера — connected (успех) или disconnected (отказ). Запрос на разъединение возможен как от сервера, так и от клиента.

После установления соединения передача ведется по дейтаграммному протоколу IPX.

Сети X.25 и Frame Relay. *Сети* X.25, работающие по одноименному стеку протоколов, предложенному международным телекоммуникационным союзом ITU (Internetional Telecommunication Union), относятся к первому поколению сетей коммутации пакетов. Протоколы X.25 разработаны еще в 1976 г. В свое время они получили широкое распространение, а в России их популярность остается значительной и в 90-е гг., поскольку эти сети хорошо приспособлены к работе на телефонных каналах невысокого качества, составляющих в России значительную долю каналов связи. С помощью сетей X.25 удобно соединять локальные сети в территориальную сеть, устанавливая между ними мосты X.25.

Стандарт X.25 относится к трем нижним уровням ЭМВОС, т.е. включает протоколы физического, канального и сетевого уровней. На сетевом уровне используется коммутация пакетов.

Характеристики сети:

— пакет содержит адресную, управляющую, информационную и контрольную части, т.е. в его заголовке имеются флаг, адреса отправителя и получателя, тип кадра (служебный или информационный), номер кадра (используется для правильной сборки сообщения из пакетов);

— на канальном уровне применено оконное управление, размер окна задает число кадров, которые можно передать до получения подтверждения (это число равно 8 или 128);

— передача данных по виртуальным (логическим) каналам, это относится к сетям с установлением соединения;

— узлы на маршруте, обнаружив ошибку, ликвидируют ошибочный пакет и запрашивает повторную передачу пакета.

В сетевом протоколе X.25 значительное внимание уделено контролю ошибок (в отличие, например, от протокола IP, в котором обеспечение надежности передается на транспортный уровень). Эта особенность приводит к уменьшению скорости передачи, т.е. сети X.25 низкоскоростные, но при этом их можно реализовать на каналах связи с невысокой помехоустойчивостью. Контроль ошибок производится при инкапсуляции и восстановлении пакетов (во всех промежуточных узлах), а не только в оконечном узле.

При использовании на физическом уровне телефонных каналов для подключения к сети достаточно иметь компьютер и модем. Подключение осуществляет провайдер (провайдерами для X.25 являются, например, владельцы ресурсов сетей Sprint, Infotel, Роспак и др.)

Типичная структура сети X.25 показана на рис. 2.10.

Типичная АКД в X.25 — синхронный модем с дуплексным бит-ориентированным протоколом. Скорости от 9,6 до 64 кбит/с. Протокол физического уровня для связи с цифровыми каналами передачи данных- X.21, а с аналоговыми — X.21bis.

Рис. 2.10. Сеть X.25

В сетях пакетной коммутации *Frame Relay (FR)* в отличие от сетей X.25 обеспечивается большая скорость передачи данных за счет исключения контроля ошибок в промежуточных узлах, так как контроль, адресация, инкапсуляция и восстановление выполняются в оконечных пунктах, т.е. на транспортном уровне. В промежуточных узлах ошибочные пакеты могут только отбрасываться, а запрос на повторную передачу происходит от конечного узла средствами уровня, выше сетевого. Но для реализации FR нужны помехоустойчивые каналы передачи данных.

Другая особенность FR — пункты доступа фиксируются при настройке порта подключения к се-

ти. Поэтому наиболее подходящая сфера применения FR — объединение совокупности ЛВС, находящихся на значительном расстоянии друг от друга.

В сетях FR сигнализация о перегрузках осуществляется вставкой соответствующих битов в заголовок пакетов, проходящих по перегруженному маршруту, управление потоками предусматривает динамическое распределение полосы пропускания между соединениями. Поэтому возможна, в отличие от сетей X.25, не только передача данных, но и передача оцифрованного голоса (так как для передачи голоса обычно требуется режим реального времени). По этой же причине FR лучше приспособлены для передачи неравномерного трафика, характерного для связей между ЛВС.

Сети FR также получают широкое распространение в России по мере развития помехоустойчивых каналов связи, так как облегчен переход к ним от сетей X.25.

Заметим, что радикальное повышение скоростей передачи интегрированной информации связывают с внедрением сетей ATM.

Сети ATM. *Технология* ATM *(Asynchronous Transfer Mode)***,** реализованная в сетях ATM**,** относится к перспективным технологиям, обеспечивающим высокие скорости передачи разнородной информации (данных, речевых и видеосигналов) на значительные расстояния. Действительно, передача голосовой и видеоинформации обычно требуется в режиме реального времени, видеоинформация характеризуется большими объемами и, следовательно, задержки должны быть только малыми (так, для голосовой связи — около 6 с).

Технология ATM представляет собой быструю коммутацию коротких пакетов фиксированной длины (53 байт), называемых *ячейками*. В силу этой причины и саму технологию ATM иногда называют коммутацией ячеек.

Сети ATM относят к сетям с установлением соединения. Соединения могут быть постоянными и динамическими. Первые устанавливаются и разрываются администратором сети, их действие продолжительно, для каждого нового обмена данными между абонентами постоянного соединения не нужно тратить время на его установление. Вторые устанавливаются и ликвидируются автоматически для каждого нового сеанса связи.

Каждое соединение получает свой идентификатор, который указывается в заголовке ячеек. При установлении соединения каждому коммутатору на выбранном пути следования данных передается таблица соответствия идентификаторов и портов коммутаторов. Коммутатор, распознав идентификатор, направляет ячейку в нужный порт. Непосредственное указание в заголовке адресов получателя и отправителя не требуется, заголовок короткий — всего 5 байтов.

Высокие скорости в ATM обеспечиваются рядом технических решений.

Во-первых, большое число каналов с временным мультиплексированием (TDM) можно использовать для параллельной передачи частей одного и того же “объемного” сообщения (*статистическое мультиплексирование*). При этом цикл синхронизации состоит из отдельных участков, длины участка и ячейки совпадают. Под конкретное сообщение можно выделить N интервалов, совокупность которых называют виртуальным каналом. Скорость передачи можно регулировать, изменяя N. Если сеть ATM оказывается перегруженной, то во избежание потери информации возможна буферизация данных для выравнивания загрузки каналов. Регулирование загрузки (управление потоком) осуществляется периодическим включением (обычно через 32 кадра) служебной *RM*-ячейки в информационный поток. В эту ячейку промежуточные коммутаторы и конечный узел могут вставлять значения управляющих битов, сигнализирующие о перегрузке или недогрузке канала. *RM*-ячейка от конечного узла передается в обратном направлении источнику сообщения, который может соответственно изменить режим передачи. В частности, применяется режим занятия всех свободных ресурсов при перегрузке. Таким образом, происходит динамическое перераспределение нагрузки.

Во-вторых, отрицательные квитанции при искажениях собственно сообщений (но не заголовков) возможны только от конечного пункта. Это исключает потери времени в промежуточных пунктах на ожидание подтверждений. Такой способ иногда называют *коммутацией кадров* (в отличие от коммутации пакетов). Контрольный код (четырехбайтный циклический) для информационной части сообщения имеется только в конце последнего пакета сообщения.

В-третьих, упрощена маршрутизация. Собственно установление соединения выполняется ана-

логично этой процедуре в TCP/IP. Однако далее номер рассчитанного маршрута помещается в заголовок каждого пакета, и для них не нужно заново определять маршрут по таблицам маршрутизаторов при прохождении через сеть. Такую передачу называют *маршрутизацией от источника*. Другими словами, осуществляется передача с установлением соединения (в отличие, например, от IP). При этом клиент направляет серверу запрос в виде специального управляющего кадра. Кадр проходит через промежуточные маршрутизаторы и (или) коммутаторы, в которых соединению (каналу) присваивается номер VCI (идентификатор) маршрута. Если передача адресована нескольким узлам, то соответствующий VCI в коммутаторах присваивается нескольким каналам.

В-четвертых, *фиксированная длина* пакетов (кадров) упрощает алгоритмы управления и буферизации данных, исключает необходимость инкапсуляции или конвертирования пакетов при смене форматов в промежуточных сетях (если они соответствуют формату ячейки АТМ).

В технологии АТМ введены три уровня (рис. 2.11). Адаптационный уровень (AAL) аналогичен транспортному уровню в ЭМВОС, на нем происходит разделение сообщения на 48-байтные ячейки, преобразование битовых входных потоков в один поток с соблюдением пропорций между числом ячеек для данных, голосовой и видеоинформации, определение вида сервиса. При этом должна поддерживаться скорость передачи данных, необходимая для обеспечения соответствующего сервиса. На следующем уровне, называемом АТМ, к каждой ячейке добавляется пятибайтовый заголовок с маршрутной информацией. Третий уровень — физический (Р — physical) — служит для преобразования данных в электрические или оптические сигналы.

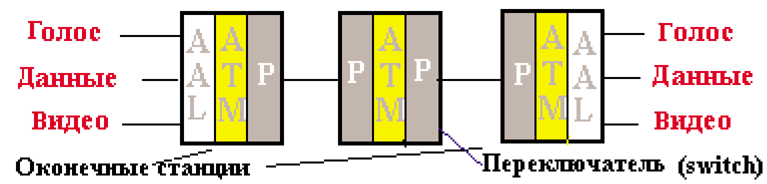


Рис. 2.11. Уровни протоколов в технологии АТМ

Средой для АТМ обычно служит среда B-ISDN, реализуемая на ВОЛС, витой паре или коаксиальном кабеле. Типично использование технологии цифровой синхронной иерархии SDH на ВОЛС. При использовании магистральной сети SDH для передачи информации по технологиям АТМ или FR сети АТМ и FR называют *наложенными вторичными сетями*. Доступ к транспортной сети осуществляется через специальные мультиплексоры.

Примером высокоскоростной магистральной сети передачи данных может служить московская сеть SDH, созданная фирмой МТУ-Информ. В 1997 г. в этой сети использовались кольцо STM-16 и три кольца STM-4, связанные между собой потоками STM-1. На периферии сети имеется 25 колец STM-1. В узлах первой очереди использованы 13 мультиплексоров SDM-16 и 59 мультиплексоров SDM-1 семейства SYNCOM, связанных ВОЛС. По каждому кольцу STM-1, STM-4, STM-16 может передаваться 63, 252 или 1008 потоков E1 соответственно, что эквивалентно 1890, 7560 или 30240 телефонным каналам. Высока надежность передачи данных, поскольку для каждого потока данных образуется два канала — основной и дублирующий, по которым одна и та же информация передается параллельно. Подключение к сети — через FR или АТМ на расстояниях до 3 км. Сеть развивается, кольца STM-4 преобразуются в STM-16, число колец растет.

Каналы АТМ со скоростями 51, 155, 622 и 2400 Мбит/с называют каналами ОС-1, ОС-3, ОС-12 и ОС-48 соответственно. К сожалению, в распространенных протоколах, таких, как TCP/IP или X.25, пакеты имеют переменную длину, что вызывает трудности совмещения программно-аппаратных средств распространенных технологий и АТМ, в связи с чем замедляется внедрение АТМ.

В настоящее время распространены также промежуточные технологии. Таковой прежде всего является технология *ретрансляции кадров* (FR), в которой применена коммутация пакетов длиной 4 Кбита с установлением соединения.

Проблемы совмещения технологий АТМ и существующих сетей решаются организацией ATM Forum и рядом промышленных фирм. Разрабатываются коммутаторы и концентраторы, обеспечивающие совместную работу АТМ-магистралей, сетей, работающих по протоколам TCP/IP, и локальных сетей, таких, как Ethernet, Fast Ethernet, FDDI. В частности, разработаны спецификации IP-over-ATM и более современные MPOA (Multi-Protocol-Over-ATM), а также реализующие их средства для передачи IP-дейтаграмм и пакетов, сформированных по другим протоколам, через АТМ-сети.

При реализации TCP/IP поверх АТМ протоколов необходимо сохранить высокую скорость АТМ сети. Однако этому препятствуют возможные потери при передаче некоторых 53-байтных ячеек, на которые разбивается TCP-сегмент. Такая потеря вызывает необходимость повторной передачи всех ячеек сегмента, поскольку в АТМ контроль правильности передачи ведется по отношению ко всему

сообщению (в данном случае — сегменту). Существенно сократить число повторно передаваемых ячеек позволяют специальные алгоритмы.

Промышленные сети. В интегрированных системах проектирования и управления на уровнях цеховом и ниже используются специальные вычислительные сети АСУТП, называемые *промышленными* (или Fieldbus). В число узлов сети входят компьютеры, выполняющие функции числового управления (NC) технологическим оборудованием и SCADA. Обычными для промышленных сетей являются предельные расстояния между узлами (датчиками, исполнительными устройствами и контроллерами) в сотни метров, размеры сообщений — до одного килобайта (в сжатой форме). Опрос датчиков периодический. Важное требование к промышленной сети — обеспечение работы в реальном масштабе времени, поэтому для АСУТП сети типа Ethernet не подходят, поскольку в них не гарантируется ограничение задержек сверху.

Пример промышленной сети — Profibus, скорость 12 Мбод, пакеты до 247 байт, расстояния до 1,5 км. Имеет выход в сеть АСУП, в качестве которой чаще всего используется сеть Ethernet. Наряду с Profibus, используют и другие протоколы, например, популярен протокол CAN. На физическом уровне в Fieldbus часто используют интерфейс RS-485 — витая пара, длина сегмента до 1,2 км, на сегменте может быть до 32 узлов.

Сетевое коммутационное оборудование. Узлы в средах передачи данных, выполняющие функции связи между частями сложной сети (internetworking), составляют *сетевое (коммутационное) оборудование*. В сетевое оборудование входят повторители, мосты, концентраторы, коммутаторы, маршрутизаторы, шлюзы, модемы и др.

Повторитель (repeater) — блок взаимодействия, служащий для регенерации электрических сигналов, передаваемых между двумя сегментами ЛВС. Повторители используются в случае, если реализация ЛВС на одном сегменте кабеля (отрезке, моноканале) не допускается из-за ограничений на расстояние или на число узлов, причем при условии, что в соседних сегментах используются один и тот же метод доступа и одни и те же протоколы. Трафик в сегментах, соединенных повторителем, — общий. Повторитель может быть многопортовым. Сигнал, пришедший на один из портов, повторяется на всех остальных портах.

Мост (bridge) — блок взаимодействия, служащий для соединения разных подсетей, которые могут иметь неодинаковые канальные протоколы.

При малых расстояниях между подсетями связь возможна через серверы подсетей, в которых размещаются интерфейсные платы, называемые внутренними мостами, и соответствующее сетевое программное обеспечение. Возможно применение внешних мостов — специально выделяемых узлов для целей сопряжения по одному в каждой из соединяемых подсетей. Внешние мосты обходятся дороже, но обеспечивают лучшие эксплуатационные характеристики. Важная функция мостов — ограничение трафика, так как локальный трафик одной подсети замыкается в ней, не проходя в другую подсеть.

Обычно мост имеет два порта, хотя существуют и многопортовые мосты. Каждый порт может оказаться входным или выходным. Управление передачей пакетов выполняется с помощью маршрутной таблицы моста, в которой строки содержат соответствующие друг другу значения адреса узла и номера порта моста. Если пакет пришел на порт A и по таблице адрес относится к тому же порту A, то пакет остается в данной ЛВС, иначе передается на порт B, который найден по таблице. Первоначальное заполнение таблицы происходит по адресам источников пакетов — в строку заносятся адрес отправителя и номер входного порта. Таблицы могут изменять во времени свое содержимое. Если некоторые адреса по истечении длительного времени ни разу не активировались, то строки с такими адресами удаляются, их восстановление или занесение новых адресов выполняется по процедуре первоначального заполнения.

На рис. 2.12 представлены возможные варианты мостовых соединений.

Вариант 2 обеспечивает большую пропускную способность по сравнению с вариантом 1. Вариант 3 близок к варианту 2 по пропускной способности, он дороже, но именно его необходимо применять, если расстояния между подсетями довольно большие. Вариант 4 используют для увеличения скорости при большом трафике. Наконец, вариант 5 предназначен для случаев расстояний в несколько километров и более.

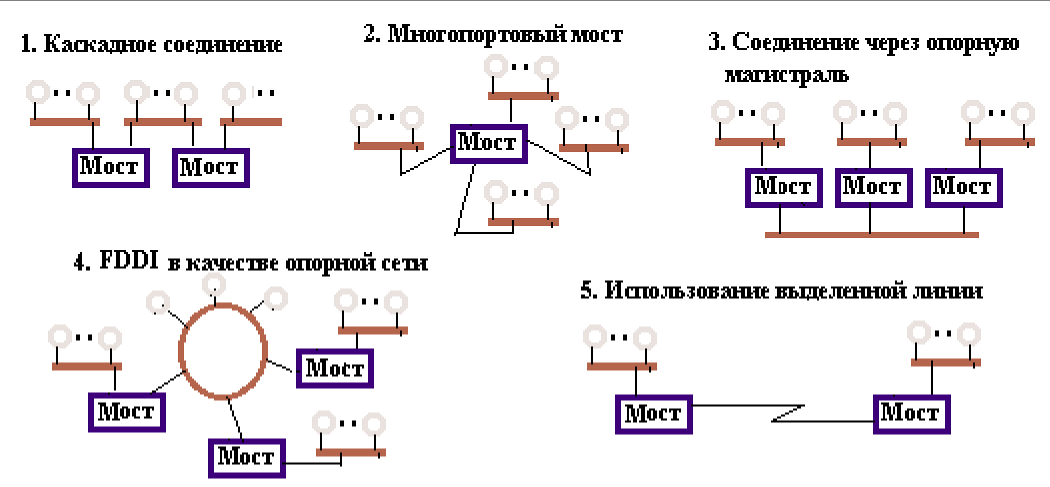


Рис. 2.12. Варианты мостовых соединений

Корпоративную сеть, состоящую из подсетей, связанных мостами, можно назвать автономной системой (AS — Autonomous System). Связь одной AS с другими осуществляется через маршрутизатор или шлюз. Такой маршрутизатор называют пограничным. В качестве AS можно рассматривать и более сложную совокупность связанных AS, если эта совокупность имеет выход во внешние сети опять же через пограничный маршрутизатор (шлюз). Из сказанного следует, что структура глобальных сетей является иерархической.

Концентраторы предназначены для объединения в сеть многих узлов. Так, концентраторами являются хабы в сетях 10Base-T или Token Ring. Однако такие концентраторы подобно мостам создают общую среду передачи данных без разделения трафика.

Коммутаторы в отличие от концентраторов предназначены для объединения в сеть многих узлов или подсетей с разделением трафика между подсетями. Как и в мостах, пакеты передаются только в ту подсеть, для которой они предназначены, что уменьшает общую загрузку сети. Но в отличие от многопортового моста в коммутаторе возможно одновременно иметь много соединений, т.е. обеспечивается параллельная передача сообщений. Коммутаторы используют также для связи нескольких ЛВС с территориальной сетью. Один коммутатор может объединять несколько как однотипных, так и разнотипных ЛВС.

Маршрутизатор (router) — блок взаимодействия, служащий для выбора маршрута передачи данных в корпоративных и территориальных сетях. С помощью маршрутизаторов могут согласовываться не только канальные протоколы, как это имеет место при применении мостов, но и сетевые протоколы. Маршрутизаторы содержат таблицы и протоколы маршрутизации в отличие от других узлов, которые могут содержать лишь локальные таблицы соответствия IP-адресов физическим адресам сетевых контроллеров в локальной сети. Маршрутизаторы могут фильтровать пакеты в соответствии с признаками, отраженными в заголовке пакета, т.е. выполнять роль брандмауэра — устройства, защищающего сеть от нежелательных вторжений извне.

Использование коммутаторов вместо маршрутизаторов (там, где это возможно) позволяет существенно повысить пропускную способность сети. Коммутатор работает с локальными MAC-адресами, в нем имеется таблица соответствия MAC-адресов и портов. Кроме того, между разными портами коммутатора образуется несколько соединений, по которым пакеты могут передаваться одновременно. В то же время маршрутизатор оперирует IP-адресами и таблицами маршрутизации и выполняет сложные алгоритмы маршрутизации.

Возможны коммутация “на лету” (сквозная коммутация — out-trough) , когда передача пакета начинается сразу после расшифровки заголовка, и после полного получения пакета (промежуточная буферизация — store-and-forward). Первый способ чаще применяют в небольших сетях, второй — в магистральных коммутаторах. Сквозная коммутация позволяет уменьшить задержки в передаче пакетов и обойтись малым объемом буфера, но не дает возможности контролировать безошибочность передачи данных.

Обычно коммутатор имеет системную плату, ряд портов, группируемых в сегменты, систему коммутации портов и функциональные модули. Каждый сегмент ориентирован на ЛВС одного типа.

Так, коммутатор ODS Infinity фирмы OpticalData Systems имеет в своем составе сегменты для сетей типов Ethernet, Token Ring, FDDI, LocalTalk, причем в этих сегментах имеются гнезда для подключения 48, 48, 2 и 6 сетей соответственно. Порты соединяются посредством высокоскоростной общей шины (что более характерно для многопортовых мостов), но чаще через коммутирующую матрицу. Функциональные модули предназначены для связи сегментов и выхода в территориальную сеть.

Различают коммутаторы второго и третьего уровней. Сети с коммутаторами второго уровня подвержены так называемому широковещательному шторму, поскольку при широковещательной передаче пакеты направляются во все подсети, соединенные через коммутаторы, и сеть будет “забита” пакетами. Чтобы уменьшить отрицательное влияние такого шторма, сеть разбивают на подсети, в пределах которых и осуществляется широковещательность. Коммутатор третьего уровня разделяет подсети, направляя через себя пакет только, если MAC-адрес получателя относится к другой подсети.

Обычно распределение узлов по подсетям выполняют по территориальному признаку. Однако при этом возможно объединение в одной подсети узлов, слабо связанных между собой в функциональном отношении. Возникают проблемы с защитой информации и управлением трафиком. Поэтому предпочтительнее распределять узлы по функциональному признаку, причем администратор сети должен иметь возможность перекоммутации узлов при изменениях в их функциях или расположении. Такие возможности имеются в виртуальных ЛВС.

Виртуальная ЛВС (ВЛВС) — это локальная сеть, в которой узлы сгруппированы не по территориальному, а по функциональному признаку. Для этого каждая подсеть в ВЛВС получает свой идентификатор, каждому идентификатору соответствуют определенные порты коммутаторов сети. Идентификатор указывается в заголовке кадра (структура кадра в ВЛВС задается стандартом IEEE 802/10) и поэтому коммутатор направляет кадр в нужную подсеть. Администратор сети может управлять структурой сети (перекоммутацией портов) с помощью специального ПО.

Лидером в производстве коммутаторов для ВЛВС является фирма Cisco. Ее коммутаторы семейства Catalyst допускают объединение в ВЛВС до 1024 подсетей FDDI, E, TR, ATM. Встроенные программы управления позволяют закреплять любой порт за любой подсетью.

Шлюз (gateway — межсетевой преобразователь) — блок взаимодействия, служащий для соединения информационных сетей различной архитектуры и с неодинаковыми протоколами. В шлюзах предусмотрено согласование протоколов всех семи уровней ЭМВОС. Примерами шлюзов могут быть устройства, соединяющие ЛВС типа Ethernet с сетью SNA, используемой для связи больших машин фирмы IBM. Часто под шлюзом понимают сервер, имеющий единственный внешний канал передачи данных.

Упражнения и вопросы для самоконтроля

1. Поясните состав и назначение устройств графической рабочей станции.
2. Что такое “растеризация” и “векторизация”?
3. Что такое “промышленный компьютер”? Каковы его особенности?
4. Дайте сравнительную характеристику методов коммутации каналов и коммутации пакетов.
5. В чем заключается сущность методов временного (TDM) и частотного (FDM) разделения каналов?
6. Почему в МДКН/ОК повторные попытки захвата линии разрешаются через случайные интервалы времени?
7. Рассчитайте размер окна столкновений в сети 10Base-5, если линия передачи данных представлена одним сегментом кабеля длиной 500 м.
8. Что такое “стаффинг”?
9. В чем сущность метода предотвращения конфликтов в RadioEthernet?
10. Почему способ кодирования 4b/5b или 8b/10b позволяет увеличить информационную скорость передачи данных?
11. Каким образом реализуется приоритетная передача данных в сети Token Ring?
12. Почему в сетях Ethernet введено ограничение на размер кадра снизу? Рассчитайте нижнюю границу длины кадра для Gigabit Ethernet.
13. Какой может быть максимальная информационная скорость в канале передачи данных с полосой пропускания 4 кГц и отношением сигнал/помеха 130?
14.В чем заключаются преимущества перевода системы сотовой связи в более высокочастотный диапазон?

15. Рассчитайте задержку в передаче сигнала в спутниковых системах с использованием геостационарных орбит (высота спутника 36 тыс. км).

16. Сколько телефонных разговоров одновременно можно передавать по каналу Т1?

17. Поясните, как действует схема эхо-компенсации.

18. Каким образом выполняется контроль правильности передачи данных по протоколу TCP?

19. Почему в IP-пакете имеется контрольный код заголовка, а не всего пакета?

20. Что такое “менеджеры” и “агенты” в сетевом программном обеспечении?

21. Назовите факторы, обусловливающие высокие скорости передачи данных в сетях ATM.

22. Что такое “маршрутизация от источника”?

23 Что понимают под виртуальной ЛВС?

Глава 3 Математическое обеспечение анализа проектных решений

3.1. Компоненты математического обеспечения

Математический аппарат в моделях разных иерархических уровней. К МО анализа относят математические модели, численные методы, алгоритмы выполнения проектных процедур. Компоненты МО определяются базовым математическим аппаратом, специфичным для каждого из иерархических уровней проектирования.

На *микроуровне* типичные математические модели (ММ) представлены дифференциальными уравнениями в частных производных (ДУЧП) вместе с краевыми условиями. К этим моделям, называемым *распределенными*, относятся многие уравнения математической физики. Объектами исследования здесь являются поля физических величин, что требуется при анализе прочности строительных сооружений или машиностроительных деталей, исследовании процессов в жидких средах, моделировании концентраций и потоков частиц и т.п.

Число совместно исследуемых различных сред (число деталей, слоев материала, фаз агрегатного состояния) в практически используемых моделях микроуровня не может быть большим из-за сложностей вычислительного характера. Резко снизить вычислительные затраты в многокомпонентных средах можно, только применив иной подход к моделированию, основанный на принятии определенных допущений.

Допущение, выражаемое дискретизацией пространства, позволяет перейти к моделям *макроуровня*. Моделями макроуровня, называемыми также *сосредоточенными*, являются системы алгебраических и обыкновенных дифференциальных уравнений, поскольку независимой переменной здесь остается только время t. Упрощение описания отдельных компонентов (деталей) позволяет исследовать модели процессов в устройствах, приборах, механических узлах, число компонентов в которых может доходить до нескольких тысяч.

В тех случаях, когда число компонентов в исследуемой системе превышает некоторый порог, сложность модели системы на макроуровне вновь становится чрезмерной. Поэтому, принимая соответствующие допущения, переходят на *функционально-логический* уровень. На этом уровне используют аппарат передаточных функций для исследования аналоговых (непрерывных) процессов или аппарат математической логики и конечных автоматов, если объектом исследования является дискретный процесс, т.е. процесс с дискретным множеством состояний.

Наконец, для исследования еще более сложных объектов, примерами которых могут служить производственные предприятия и их объединения, вычислительные системы и сети, социальные системы и другие подобные объекты, применяют аппарат теории массового обслуживания, возможно использование и некоторых других подходов, например, сетей Петри. Эти модели относятся к *системному* уровню моделирования.

Требования к математическим моделям и численным методам в САПР. Основными требованиями к математическим моделям являются требования адекватности, точности, экономичности.

Модель всегда лишь приближенно отражает некоторые свойства объекта. *Адекватность* имеет место, если модель отражает заданные свойства объекта с приемлемой точностью. Под *точностью* понимают степень соответствия оценок одноименных свойств объекта и модели.

Экономичность (*вычислительная эффективность*) определяется затратами ресурсов, требуемых для реализации модели. Поскольку в САПР используются математические модели, далее речь пойдет о характеристиках именно математических моделей, и экономичность будет характеризоваться затратами машинных времени и памяти.

Адекватность оценивается перечнем отражаемых свойств и областями адекватности. *Область адекватности* — область в пространстве параметров, в пределах которой погрешности модели остаются в допустимых пределах. Например, область адекватности линеаризованной модели поверхности детали определяется системой неравенств

$$\max |\varepsilon_{ij}| \le \varepsilon_{доп},$$

где $\varepsilon_{ij} = (x_{ij\ ист} - x_{ij\ мод}) / x_{ij\ ист}$, $x_{ij\ ист}$ и $x_{ij\ мод}$ — i-я координата j-й точки поверхности в объекте и модели соответственно, ε_{ij} и $\varepsilon_{доп}$ — допущенная и предельно допустимая относительные погрешности моделирования поверхности, максимум берется по всем координатам и контролируемым точкам.

Отметим, что в большинстве случаев области адекватности строятся в пространстве внешних переменных. Так, область адекватности модели электронного радиоэлемента обычно выражает допустимые для применения модели диапазоны изменения моделируемых температур, внешних напряжений, частот.

Аналогичные требования по точности и экономичности фигурируют при выборе численных методов решения уравнений модели.

Место процедур формирования моделей в маршрутах проектирования. Вычислительный процесс при анализе состоит из этапов формирования модели и ее исследования (решения). В свою очередь, формирование модели включает две процедуры: во-первых, разработку моделей отдельных компонентов, во-вторых, формирование модели системы из моделей компонентов.

Первая из этих процедур выполняется предварительно по отношению к типовым компонентам вне маршрута проектирования конкретных объектов. Как правило, модели компонентов разрабатываются специалистами в прикладных областях, причем знающими требования к моделям и формам их представления в САПР. Обычно в помощь разработчику моделей в САПР предлагаются методики и вспомогательные средства, например, в виде программ анализа для экспериментальной отработки моделей. Созданные модели включаются в библиотеки моделей прикладных программ анализа.

На маршруте проектирования каждого нового объекта выполняется вторая процедура (рис. 3.1) — формирование модели системы с использованием библиотечных моделей компонентов. Как правило, эта процедура выполняется автоматически по алгоритмам, включенным в заранее разработанные программы анализа. Примеры таких программ имеются в различных приложениях и прежде всего в отраслях общего машиностроения и радиоэлектроники.

При применении этих программ пользователь описывает исследуемый объект на входном языке программы анализа не в виде системы уравнений, которая будет получена автоматически, а в виде списка элементов структуры, эквивалентной схемы, эскиза или чертежа конструкции.

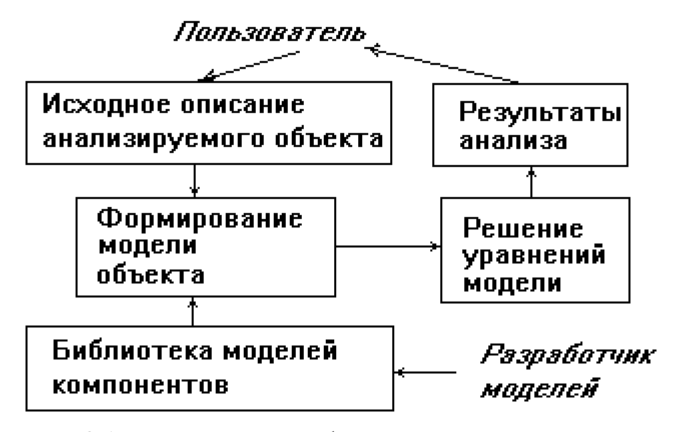


Рис. 3.1. Место процедур формирования моделей на маршрутах проектирования

3.2. Математические модели в процедурах анализа на макроуровне

Исходные уравнения моделей. Исходное математическое описание процессов в объектах на макроуровне представлено системами обыкновенных дифференциальных и алгебраических уравнений. Аналитические решения таких систем при типичных значениях их порядков в практических задачах получить не удается, поэтому в САПР преимущественно используются алгоритмические модели. В этом параграфе изложен обобщенный подход к формированию алгоритмических моделей на макроуровне, справедливый для большинства приложений.

Исходными для формирования математических моделей объектов на макроуровне являются компонентные и топологические уравнения.

Компонентными уравнениями называют уравнения, описывающие свойства элементов (компонентов), другими словами, это уравнения *математических моделей элементов* (ММЭ).

Топологические уравнения описывают взаимосвязи в составе моделируемой системы.

В совокупности компонентные и топологические уравнения конкретной физической системы представляют собой исходную *математическую модель системы* (ММС).

Очевидно, что компонентные и топологические уравнения в системах различной физической природы отражают разные физические свойства, но могут иметь одинаковый формальный вид. Одинаковая форма записи математических соотношений позволяет говорить о *формальных аналогиях*

компонентных и топологических уравнений. Такие аналогии существуют для механических поступательных, механических вращательных, электрических, гидравлических (пневматических), тепловых объектов. Наличие аналогий приводит к практически важному выводу: *значительная часть алгоритмов формирования и исследования моделей в САПР оказывается инвариантной и может быть применена к анализу проектируемых объектов в разных предметных областях.* Единство математического аппарата формирования ММС особенно удобно при анализе систем, состоящих из физически разнородных подсистем.

В перечисленных выше приложениях компонентные уравнения имеют вид

$$\mathbf{F}_к(d\mathbf{V}/dt, \mathbf{V}, t) = 0 \qquad (3.1)$$

и топологические уравнения

$$\mathbf{F}_т(\mathbf{V}) = 0, \qquad (3.2)$$

где $\mathbf{V} = (v_1, v_2, \dots v_n)$ — вектор фазовых переменных, t — время.

Различают фазовые переменные двух типов, их обобщенные наименования — фазовые переменные типа потенциала (например, электрическое напряжение) и типа потока (например, электрический ток). Каждое компонентное уравнение характеризует связи между разнотипными фазовыми переменными, относящимися к одному компоненту (например, закон Ома описывает связь между напряжением и током в резисторе), а топологическое уравнение — связи между однотипными фазовыми переменными в разных компонентах.

Модели можно представлять в виде систем уравнений или в графической форме, если между этими формами установлено взаимно однозначное соответствие. В качестве графической формы часто используют эквивалентные схемы.

Примеры компонентных и топологических уравнений. Рассмотрим несколько типов систем.

Электрические системы. В электрических системах фазовыми переменными являются электрические напряжения и токи. Компонентами систем могут быть простые двухполюсные элементы и более сложные двух- и многополюсные компоненты. К простым двухполюсникам относятся следующие элементы: сопротивление, емкость и индуктивность, характеризуемые одноименными параметрами *R, C, L*. В эквивалентных схемах эти элементы обозначают в соответствии с рис. 3.2,*а*.

Компонентные уравнения простых двухполюсников:

для R: $u = i R$ (закон Ома), (3.3)

для C: $i = C\, du/dt$, (3.4)

для L: $u = L\, di/dt$, (3.5)

где u — напряжение (точнее, падение напряжения на двухполюснике), i — ток.

Эти модели лежат в основе моделей других возможных более сложных компонентов. Большая сложность может определяться нелинейностью уравнений (3.3) — (3.5) (т.е. зависимостью *R, C, L* от фазовых переменных), или учетом зависимостей параметров *R, C, L* от температуры, или наличием более двух полюсов. Однако многополюсные компоненты могут быть сведены к совокупности взаимосвязанных простых элементов.

Топологические уравнения выражают законы Кирхгофа для напряжений (ЗНК) и токов (ЗТК). Согласно ЗНК, сумма напряжений на компонентах вдоль любого замкнутого контура в эквивалентной схеме равна нулю, а в соответствии с ЗТК сумма токов в любом замкнутом сечении эквивалентной схемы равна нулю:

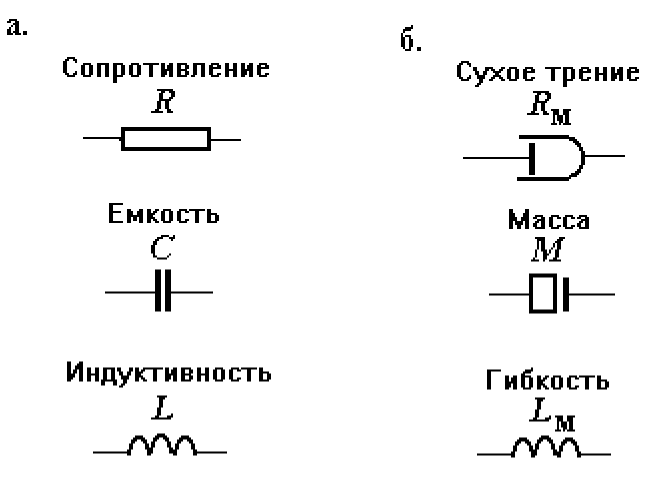


Рис. 3.2. Условные обозначения простых элементов в эквивалентных схемах: **а)** электрических, гидравлических, тепловых; **б)** механических

$$\sum_{k \in \mathbf{K}_p} u_k = 0, \qquad (3.6)$$

$$\sum_{j \in \mathbf{J}_q} i_j = 0, \qquad (3.7)$$

где $\mathbf{K}_p$ — множество номеров элементов p-го контура, $\mathbf{J}_q$ — множество номеров элементов, входящих в q-е сечение.

Примером ММ сложного компонента может служить модель транзистора. На рис. 3.3 представлена эквивалентная схема биполярного транзистора, на которой зависимые от напряжений источники тока $i_{эд} = i_{тэ}\exp(u_э/(m\varphi_т))$ и $i_{кд} = i_{тк}\exp(u_к/(m\varphi_т))$ отображают статические вольтамперные характеристики *p-n* переходов, $i_{тэ}$ и $i_{тк}$ — тепловые токи переходов, $m\varphi_т$ — температурный потенциал, $u_э$ и $u_к$ — напряжения на эмиттерном и коллекторном переходах, $C_э$ и $C_к$ — емкости переходов, $R_{уэ}$ и $R_{ук}$ — сопротивления утечки переходов, $R_б$ и $R_к$ — объемные сопротивления тел базы и коллектора, $i_г = Bi_{эд} — B_и i_{кд}$ — источник тока, моделирующий усилительные свойства транзистора, B и $B_и$ — прямой и инверсный коэффициенты усиления тока базы. Здесь $u_э$, $u_к$, $i_{эд}$, $i_{кд}$, $i_г$ — фазовые переменные, а остальные величины — параметры модели транзистора.

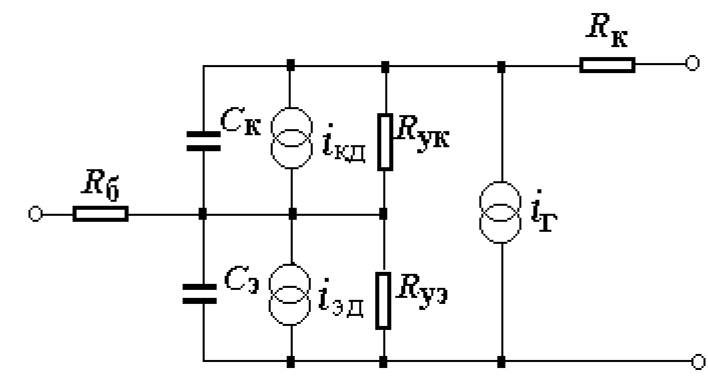


Рис. 3.3. Эквивалентная схема биполярного транзистора

Механические системы. Фазовыми переменными в механических поступательных системах являются силы и скорости. Используют одну из двух возможных электромеханических аналогий. В дальнейшем будем использовать ту из них, в которой скорость относят к фазовым переменным типа потенциала, а силу считают фазовой переменной типа потока. Учитывая формальный характер подобных аналогий, в равной мере можно применять и противоположную терминологию.

Компонентное уравнение, характеризующее инерционные свойства тел, в силу второго закона Ньютона имеет вид

$$F = M\, du / dt, \qquad (3.8)$$

где F – сила; M – масса; u — поступательная скорость.

Упругие свойства тел описываются компонентным уравнением, которое можно получить из уравнения закона Гука. В одномерном случае (если рассматриваются продольные деформации упругого стержня)

$$G = E\,\varepsilon, \qquad (3.9)$$

где G — механическое напряжение; E — модуль упругости; $\varepsilon = \Delta l/l$ — относительная деформация; Δl — изменение длины l упругого тела под воздействием G. Учитывая, что $G = F/S$, где F — сила, S — площадь поперечного сечения тела, и дифференцируя (3.9), имеем

$$dF/dt = (S\, E_ю/l)\; d(\Delta l)/dt$$

или

$$dF/dt = g\, u, \qquad (3.10)$$

где $g = (SE/l)$- жесткость (величину, обратную жесткости, называют гибкостью $L_м$); $u = d(\Delta l)/dt$ — скорость.

Диссипативные свойства в механических системах твердых тел выражаются соотношениями, характеризующими связь между силой трения и скоростью взаимного перемещения трущихся тел, причем в этих соотношениях производные сил или скоростей не фигурируют, как и в случае описания с помощью закона Ома диссипативных свойств в электрических системах.

Топологические уравнения характеризуют, во-первых, закон равновесия сил: сумма сил, приложенных к телу, включая силу инерции, равна нулю (принцип Даламбера), во-вторых, закон скоростей, согласно которому сумма относительной, переносной и абсолютной скоростей равна нулю.

В механических вращательных системах справедливы компонентные и топологические уравнения поступательных систем с заменой поступательных скоростей на угловые, сил — на вращательные

моменты, масс — на моменты инерции, жесткостей — на вращательные жесткости.

Условные обозначения простых элементов механической системы показаны на рис. 3.2,*б*.

Нетрудно заметить наличие аналогий между электрической и механической системами. Так, токам и напряжениям в первой из них соответствуют силы (либо моменты) и скорости механической системы, компонентным уравнениям (3.4) и (3.5) и фигурирующим в них параметрам C и L — уравнения (3.8) и (3.10) и параметры M и $L_м$, очевидна аналогия и между топологическими уравнениями. Далее параметры C и M будем называть емкостными (емкостного типа), параметры L и $L_м$ — индуктивными (индуктивного типа), а параметры R и $R_{тр} = \partial u/\partial F$ — резистивными (резистивного типа).

Имеется и существенное отличие в моделировании электрических и механических систем: первые из них одномерны, а процессы во вторых часто приходится рассматривать в двух- (2D) или трехмерном (3D) пространстве. Следовательно, при моделировании механических систем в общем случае в пространстве 3D нужно использовать векторное представление фазовых переменных, каждая из которых имеет шесть составляющих, соответствующих шести степеням свободы.

Однако отмеченные выше аналогии остаются справедливыми, если их относить к проекциям сил и скоростей на каждую пространственную ось, а при графическом представлении моделей использовать шесть эквивалентных схем — три для поступательных составляющих и три для вращательных.

Гидравлические системы. Фазовыми переменными в гидравлических системах являются расходы и давления. Как и в предыдущем случае, компонентные уравнения описывают свойства жидкости рассеивать или накапливать энергию.

Рассмотрим компонентные уравнения для жидкости на линейном участке трубопровода длиной Δl и воспользуемся уравнением Навье-Стокса в следующей его форме (для ламинарного течения жидкости)

$$\rho\, \partial U/\partial t = - \partial P/\partial x - 2aU,$$

где ρ — плотность жидкости; U – скорость; P – давление; a — коэффициент линеаризованного вязкого трения. Так как $U = Q/S$, где Q — объемный расход; S — площадь поперечного сечения трубопровода, то, заменяя пространственную производную отношением конечных разностей, имеем

$$dQ/dt = S / (\Delta l \rho)\Delta P - (2a/ \rho)\, Q,$$

или

$$\Delta P = L_г dQ/dt + R_г Q, \qquad (3.11)$$

где ΔP — падение давления на рассматриваемом участке трубопровода. $L_г = \Delta l\rho/S$ — гидравлическая индуктивность, отражающая инерционные свойства жидкости, $R_г = 2a/\rho$ — гидравлическое сопротивление, отражающее вязкое трение.

Примечание. В трубопроводе круглого сечения радиусом r удобно использовать выражение для гидравлического сопротивления при ламинарном течении: $R_г = 8\upsilon\Delta l/(\pi r^4)$, где υ — кинематическая вязкость; в случае турбулентного характера течения жидкости компонентное уравнение для вязкого трения имеет вид $\Delta P = R_г Q|Q|$ при $R_г = 0{,}37(\pi r\upsilon /|Q|)^{1/4}$.

Интерпретация уравнения (3.11) приводит к эквивалентной схеме рис. 3.4.

Явление сжимаемости жидкости описывается компонентным уравнением, вытекающим из закона Гука

$$\Delta P = E\Delta l/l\ . \qquad (3.12)$$

Рис. 3.4. Эквивалентная схема участка трубопровода

Дифференцируя (3.12) и учитывая, что объемный расход Q связан со скоростью $U = d(\Delta l)/dt$ соотношением $Q = U\, S$, получаем

$$d\Delta P/dt = C_г Q,$$

где $C_г = E/(S\Delta l)$ — гидравлическая емкость.

Связь подсистем различной физической природы. Используют следующие способы моделирования взаимосвязей подсистем: с помощью трансформаторной, гираторной связей и с помощью зависимости параметров компонентов одной подсистемы от фазовых переменных другой. В эквивалентных схемах трансформаторные и гираторные связи представлены зависимыми источниками фазовых переменных, показанными на рис. 3.5. На этом рисунке k и n — коэффициенты трансформации; g — пе-

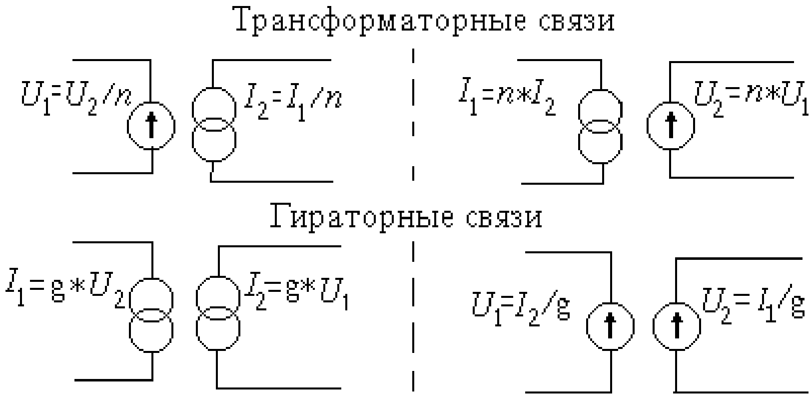


Рис. 3.5. Элементы взаимосвязи подсистем различной физической природы

редаточная проводимость; U_j и I_j – фазовые переменные в j-й цепи; j=1 соответствует первичной, а j=2 — вторичной цепи.

Представление топологических уравнений. Известен ряд методов формирования ММС на макроуровне. Получаемые с их помощью модели различаются ориентацией на те или иные численные методы решения и набором *базисных переменных*, т.е. фазовых переменных, остающихся в уравнениях итоговой ММС. Общим для всех методов является исходная совокупность топологических и компонентных уравнений (3.1)-(3.2).

При записи топологических уравнений удобно использовать промежуточную графическую форму — представление модели в виде эквивалентной схемы, состоящей из двухполюсных элементов. Общность подхода при этом сохраняется, так как любой многополюсный компонент можно заменить подсхемой из двухполюсников. В свою очередь эквивалентную схему можно рассматривать как направленный граф, дуги которого соответствуют ветвям схемы. Направления потоков в ветвях выбираются произвольно (если реальное направление при моделировании окажется противоположным, то это приведет лишь к отрицательным численным значениям потока).

Пример некоторой простой эквивалентной схемы и соответствующего ей графа приведен на рис. 3.6. Для конкретности и простоты изложения на рис. 3.6 использованы условные обозначения, характерные для электрических эквивалентных схем, по той же причине далее в этом параграфе часто применяется электрическая терминология. Очевидно, что поясненные выше аналогии позволяют при необходимости легко перейти к обозначениям и терминам, привычным для механиков.

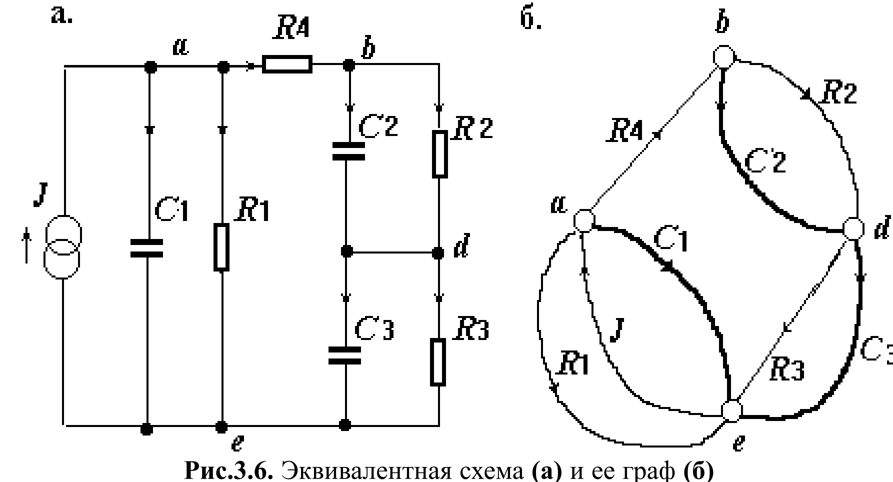


Рис.3.6. Эквивалентная схема **(а)** и ее граф **(б)**

Для получения топологических уравнений все ветви эквивалентной схемы разделяют на подмножества хорд и ветвей дерева. Имеется в виду *покрывающее* (фундаментальное) дерево, т.е. подмножество из β-1 дуг, не образующее ни одного замкнутого контура, где β — число вершин графа (узлов эквивалентной схемы). На рис. 3.6,*б* показан граф эквивалентной схемы рис. 3.6,*а*, толстыми линиями выделено одно из возможных покрывающих деревьев.

Выбор дерева однозначно определяет вектора напряжений $\mathbf{U}_х$ и токов $\mathbf{I}_х$ хорд, напряжений $\mathbf{U}_{вд}$ и токов $\mathbf{I}_{вд}$ ветвей дерева и приводит к записи топологических уравнений в виде

$$\mathbf{U}_х + \mathbf{M}\mathbf{U}_{вд} = 0, \qquad (3.13)$$

$$\mathbf{I}_{вд} - \mathbf{M}^{Т}\mathbf{I}_х = 0, \qquad (3.14)$$

где **M** — матрица контуров и сечений, $\mathbf{M}^{Т}$ — транспонированная **M**-матрица.

Таблица 3.1.

Хорды	Ветви дерева		
	*C*1	*C*2	*C*3
*R*1	-1	0	0
*R*2	0	-1	0
*R*3	0	0	-1
*R*4	-1	+1	+1
J	+1	0	0

В **M**-матрице число строк соответствует числу хорд, число столбцов равно числу ветвей дерева. **M**-матрица формируется следующим образом. Поочередно к дереву подключаются хорды. Если при подключении к дереву p-й хорды q-я ветвь входит в образовавшийся контур, то элемент M_{pq} матрицы равен +1 при совпадении

направлений ветви и подключенной хорды, M_{pq} = -1 при несовпадении направлений. В противном случае M_{pq} = 0.

Для схемы на рис. 3.6 **М**-матрица представлена в виде табл. 3.1

Особенности эквивалентных схем механических объектов. Для каждой степени свободы строят свою эквивалентную схему. Каждому телу с учитываемой массой соответствует узел схемы (вершина графа). Один узел, называемый базовым, отводится телу, отождествляемому с инерциальной системой отсчета.

Каждый элемент массы изображают ветвью, соединяющей узел соответствующего массе тела с базовым узлом; каждый элемент упругости — ветвью, соединяющей узлы тел, связанных упругой связью; каждый элемент трения — ветвью, соединяющей узлы трущихся тел. Внешние воздействия моделируются источниками сил и скоростей.

В качестве примера на рис. 3.7,*а* изображена некоторая механическая система — тележка, движущаяся по дороге и состоящая из платформы *A*, колес *B*1, *B*2 и рессор *C*1, *C*2. На рис. 3.7,*б* приведена эквивалентная схема для вертикальных составляющих сил и скоростей, на которой телам системы соответствуют одноименные узлы, учитываются массы платформы и колес, упругость рессор, трение между колесами и дорогой; неровности дороги вызывают воздействие на систему, изображенное на рис. 3.7,*б* источниками силы.

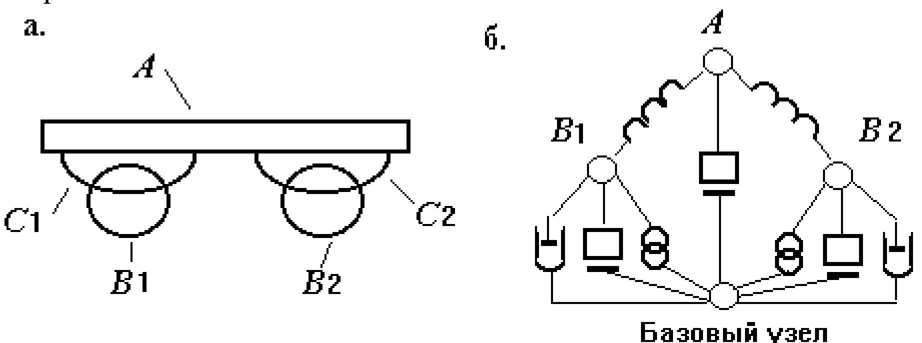


Рис. 3.7. Простая механическая система:
а - эскизное изображение; **б** - эквивалентная схема

Характеристика методов формирования ММС. Исходную систему компонентных и топологических уравнений (3.1) и (3.2) можно рассматривать как окончательную ММС, которая и подлежит численному решению. Численное решение этой системы уравнений предполагает *алгебраизацию* дифференциальных уравнений, например, с помощью преобразования Лапласа или формул численного интегрирования. В программах анализа нелинейных объектов на макроуровне, как правило, применяются формулы численного интегрирования, примером которых может служить неявная формула Эйлера

$$d\mathbf{V}/dt\,|_n = (\mathbf{V}_n - \mathbf{V}_{n-1}) / h_n,$$

где $\mathbf{V}_i$ — значение переменной **V** на *i*-м шаге интегрирования; $h_n = t_n - t_{n-1}$ — шаг интегрирования. Алгебраизация подразумевает предварительную дискретизацию независимой переменной *t* (вместо непрерывной переменной *t* получаем конечное множество значений t_n), она заключается в представлении ММС в виде системы уравнений

$$\begin{aligned} &\mathbf{F}_к(\mathbf{Z}_n, \mathbf{V}_n, t_n) = 0, \\ &\mathbf{F}_т(\mathbf{V}_n) = 0, \qquad (3.15) \\ &\mathbf{Z}_n = (\mathbf{V}_n - \mathbf{V}_{n-1}) / h_n \end{aligned}$$

с неизвестными $\mathbf{V}_n$ и $\mathbf{Z}_n$, где использовано обозначение $\mathbf{Z} = d\mathbf{V}/dt$. Эту систему алгебраических уравнений, в общем случае нелинейных, необходимо решать на каждом шаге численного интегрирования исходных дифференциальных уравнений.

Однако порядок этой системы довольно высок и примерно равен $2\alpha+\gamma$, где α — число ветвей эквивалентной схемы (каждая ветвь дает две неизвестные величины — фазовые переменные типа потока и типа потенциала, за исключением ветвей внешних источников, у каждой из которых неизвестна лишь одна фазовая переменная), γ — число элементов в векторе производных. Чтобы снизить порядок системы уравнений и тем самым повысить вычислительную эффективность ММС, желательно выполнить предварительное преобразование модели (в символическом виде) перед ее многошаговым численным решением. Предварительное преобразование сводится к исключению из системы части

неизвестных и соответствующего числа уравнений. Оставшиеся неизвестные называют *базисными*. В зависимости от набора базисных неизвестных различают несколько методов формирования ММС.

Согласно *методу переменных состояния* (более полное название метода — метод переменных, характеризующих состояние), вектор базисных переменных **W** состоит из *переменных состояния*. Этот вектор включает неизбыточное множество переменных, характеризующих накопленную в системе энергию. Например, такими переменными могут быть скорости тел (кинетическая энергия определяется скоростью, так как равна $Mu^2/2$), емкостные напряжения, индуктивные токи и т.п. Очевидно, что число уравнений не превышает γ. Кроме того, итоговая форма ММС оказывается приближенной к явной форме представления системы дифференциальных уравнений, т.е. к форме, в которой вектор $d\mathbf{W}/dt$ явно выражен через вектор **W**, что упрощает дальнейшее применение явных методов численного интегрирования. Метод реализуется путем особого выбора системы хорд и ветвей дерева при формировании топологических уравнений. Поскольку явные методы численного интегрирования дифференциальных уравнений не нашли широкого применения в программах анализа, то метод переменных состояния также теряет актуальность и его применение оказывается довольно редким.

В классическом варианте *узлового метода* в качестве базисных переменных используются *узловые потенциалы* (т.е. скорости тел относительно инерциальной системы отсчета, абсолютные температуры, перепады давления между моделируемой и внешней средой, электрические потенциалы относительно базового узла). Число узловых потенциалов и соответственно уравнений в ММС оказывается равным β-1, где β — число узлов в эквивалентной схеме. Обычно β заметно меньше α и, следовательно, порядок системы уравнений в ММС снижен более чем в два раза по сравнению с порядком исходной системы.

Однако классический вариант узлового метода имеет ограничения на применение и потому в современных программах анализа наибольшее распространение получил *модифицированный узловой метод*.

Узловой метод. Матрицу контуров и сечений **M** в узловом методе формируют следующим образом. Выбирают базовый узел эквивалентной схемы и каждый из остальных узлов соединяют с базовым фиктивной ветвью. Именно фиктивные ветви принимают в качестве ветвей дерева, а все реальные ветви оказываются в числе хорд. Поскольку токи фиктивных ветвей равны нулю, а вектор напряжений фиктивных ветвей есть вектор узловых потенциалов **φ**, то уравнения (3.13) и (3.14) принимают вид

$$\mathbf{U} + \mathbf{M}\boldsymbol{\varphi} = 0, \qquad (3.16)$$

$$\mathbf{M}^{T}\,\mathbf{I} = 0, \qquad (3.17)$$

где **U** и **I**- векторы напряжений и токов реальных ветвей.

Компонентные уравнения алгебраизуются с помощью одной из формул численного интегрирования, линеаризуются с помощью разложения в ряд Тейлора с сохранением только линейных членов, и их представляют в виде

$$\mathbf{I}_n = \mathbf{G}_n\mathbf{U}_n + \mathbf{A}_n, \qquad (3.18)$$

где $\mathbf{G}_n$ — диагональная матрица проводимостей, рассчитанная в точке t_n; $\mathbf{A}_n$ — вектор, зависящий от значений фазовых переменных на предшествующих шагах интегрирования и потому уже известный к моменту времени t_n. Каждая ветвь (за исключением идеальных источников напряжения) имеет проводимость, которая занимает одну из диагональных клеток матрицы проводимостей.

Окончательно ММС получаем, подставляя (3.18) и затем (3.16) в (3.17):

$$\mathbf{M}^{T}\mathbf{I}_n = \mathbf{M}^{T}(\mathbf{G}_n\mathbf{U}_n + \mathbf{A}_n) = -\,\mathbf{M}^{T}\mathbf{G}_n\mathbf{M}\boldsymbol{\varphi}_n + \mathbf{M}^{T}\mathbf{A}_n = 0$$

или

$$\mathbf{Я}_n\boldsymbol{\varphi}_n = \mathbf{B}_n, \qquad (3.19)$$

где $\mathbf{Я}_n = \mathbf{M}^{T}\,\mathbf{G}_n\mathbf{M}$ — матрица Якоби, $\mathbf{B}_n = \mathbf{M}^{T}\mathbf{A}_n$ — вектор правых частей. Отметим, что матрица **M** имеет размер равен α × (β-1), матрица $\mathbf{G}_n$ — α × α, а матрица Якоби — (β-1) × (β-1).

Система (3.19) является *системой линейных алгебраических уравнений*, полученной в результа-

те дискретизации независимой переменной, алгебраизации дифференциальных уравнений и линеаризации алгебраических уравнений. Алгебраизация приводит к необходимости пошагового вычислительного процесса интегрирования, линеаризация — к выполнению итерационного вычислительного процесса на каждом шаге интегрирования.

Рассмотрим, каким образом определяются проводимости ветвей.

Для резистивных ветвей проводимость — величина, обратная сопротивлению R.

При использовании неявного метода Эйлера проводимость емкостной ветви получается из ее компонентного уравнения следующим образом.

На n-м шаге интегрирования

$$i_n = Cdu/dt\,|_n = C(u_n - u_{n-1}) / h_n,$$

проводимость $g = \partial i_n / \partial u_n$ и при C = const получаем

$$g = C / h_n.$$

При этом в вектор правых частей входит элемент $a_n = g u_{n-1}$.

Проводимость индуктивной ветви можно найти аналогично:

$$u_n = L(i_n - i_{n-1}) / h_n$$

и при L = const

$$g = h_n / L,\;\; a_n = i_{n-1}.$$

Аналогично определяют проводимости и при использовании других разностных формул численного интегрирования, общий вид которых

$$d\mathbf{U}/dt\,|_n = \mu_n\, \mathbf{U}_n - \eta_n,$$

где μ_n зависит от шага интегрирования, η_n — от значений вектора $\mathbf{U}$ на предыдущих шагах.

Классический вариант узлового метода имеет ограничения на применение. Так, недопустимы идеальные (с бесконечной проводимостью) источники напряжения, зависимые источники, аргументами которых являются токи, а также индуктивности, поскольку в классическом варианте токи не входят в число базисных переменных. Устранить эти ограничения довольно просто — нужно расширить совокупность базисных координат, включив в нее токи-аргументы зависимых источников, а также токи ветвей индуктивных и источников напряжения. Полученный вариант метода называют *модифицированным узловым методом*.

Согласно модифицированному узловому методу, в дерево при построении матрицы **M** включают ветви источников напряжения и затем фиктивные ветви. В результате матрица **M** принимает вид (табл. 3.2), где введены обозначения: $\mathbf{U}_{ист}(\mathbf{I})$ — источники напряжения, зависящие от тока; $\mathbf{E}(t)$ — независимые источники напряжения; $\mathbf{I}_{ист}(\mathbf{I})$ — источники тока, зависящие от тока; L — индуктивные ветви; $\mathbf{M}_{ij}$ — подматрица контуров хорд группы i и сечений фиктивных ветвей группы j.

Таблица 3.2

Тип ветви	Фиктивные ветви	$U_{ист}(I)$	E(t)
неособые ветви	M_{11}	M_{12}	M_{13}
L	M_{21}	M_{22}	M_{23}
$I_{ист}(I)$	M_{31}	M_{32}	M_{33}

Те же обозначения $\mathbf{U}_{ист}$, **I**, **E**, $\mathbf{I}_{ист}$ будем использовать и для соответствующих векторов напряжений и токов. Назовем ветви, токи которых являются аргументами в выражениях для зависимых источников, т.е. входят в вектор **I**, *особыми* ветвями. Остальные ветви (за исключением индуктивных) — *неособые*. Введем также обозначения: $\mathbf{I}_L$ — вектор индуктивных токов; $\mathbf{I}_x$ и $\mathbf{U}_x$ — векторы токов и напряжений неособых ветвей; $\mathbf{G}_x$, $\mathbf{G}_L$, $\mathbf{G}_\mathbf{I}$ — диагональные матрицы проводимостей ветвей неособых, индуктивных, особых.

Уравнение закона токов Кирхгофа (3.17) для фиктивных ветвей имеет вид

$$(\mathbf{M}_{11})^T \mathbf{I}_x + (\mathbf{M}_{21})^T \mathbf{I}_L + (\mathbf{M}_{31})^T \mathbf{I}_{ист} = 0.$$

Исключим вектор $\mathbf{I}_x$ с помощью компонентного уравнения (3.18), а вектор $\mathbf{I}_{ист}$ с помощью оче-

видного выражения

$$\mathbf{I}_{ист} = \mathbf{KI},$$

где $\mathbf{K} = (\partial \mathbf{I}_{ист}/\partial \mathbf{I})$ — матрица передаточных коэффициентов источников тока. Используем также выражение (3.16), принимающее вид

$$\mathbf{U}_x = - \mathbf{M}_{11}\boldsymbol{\varphi} - \mathbf{M}_{12}\mathbf{U}_{ист} - \mathbf{M}_{31}\mathbf{E} = - \mathbf{M}_{11}\boldsymbol{\varphi} - \mathbf{M}_{12}\,(\partial \mathbf{U}_{ист}/\partial \mathbf{I})\,\mathbf{I} - \mathbf{M}_{31}\mathbf{E}$$

Получаем систему из трех матричных уравнений с неизвестными векторами $\boldsymbol{\varphi}$, $\mathbf{I}$ и $\mathbf{I}_L$:

$$-(\mathbf{M}_{11})^T\mathbf{G}_x(\mathbf{M}_{11}\boldsymbol{\varphi} + \mathbf{M}_{12}\mathbf{RI}) + (\mathbf{M}_{21})^T\mathbf{I}_L + (\mathbf{M}_{31})^T\mathbf{KI} = \mathbf{G}_x\mathbf{M}_{31}\mathbf{E} + (\mathbf{M}_{11})^T\mathbf{A}_x; \quad (3.20)$$

$$\mathbf{I}_L = - \mathbf{G}_L(\mathbf{M}_{21}\boldsymbol{\varphi} + \mathbf{M}_{22}\mathbf{RI} + \mathbf{M}_{23}\mathbf{E}) + \mathbf{A}_L; \quad (3.21)$$

$$\mathbf{I} = - \mathbf{G}_\mathbf{I}(\mathbf{M}_{31}\boldsymbol{\varphi} + \mathbf{M}_{32}\mathbf{RI} + \mathbf{M}_{33}\mathbf{E}) + \mathbf{A}_\mathbf{I}, \quad (3.22)$$

где обозначено $\mathbf{R} = (\partial \mathbf{U}_{ист}/\partial \mathbf{I})$. Эта система и является итоговой ММ в узловом модифицированном методе.

Замечания:

1. Вектор индуктивных токов нельзя исключить из итоговой системы уравнений, так как его значения входят в вектор $\mathbf{A}_L$ на последующих шагах численного интегрирования.

2. Источники тока, зависящие от напряжений, относятся к неособым ветвям, их проводимости $(\partial \mathbf{I}_{ист}/\partial \mathbf{U})$ входят в матрицу $\mathbf{G}_x$, которая при этом может иметь недиагональный вид.

3. Источники напряжения, зависящие от напряжений, в приведенных выше выражениях не учитываются, при их наличии нужно в матрице **M** выделить столбец для этих ветвей, что приводит к появлению дополнительных слагаемых в правых частях уравнений (3.19) — (3.21).

3.3. Методы и алгоритмы анализа на макроуровне

Выбор методов анализа во временной области. Анализ процессов в проектируемых объектах можно производить во временной и частотной областях. *Анализ во временной области* (динамический анализ) позволяет получить картину переходных процессов, оценить динамические свойства объекта, он является важной процедурой при исследовании как линейных, так и нелинейных систем. *Анализ в частотной области* более специфичен, его применяют, как правило, к объектам с линеаризуемыми ММ при исследовании колебательных стационарных процессов, анализе устойчивости, расчете искажений информации, представляемой спектральными составляющими сигналов, и т.п.

Методы анализа во временной области, используемые в универсальных программах анализа в САПР, — это численные методы интегрирования систем обыкновенных дифференциальных уравнений (СОДУ):

$$\mathbf{F}(d\mathbf{V}/dt, \mathbf{V}, t) = 0.$$

Другими словами, это методы алгебраизации дифференциальных уравнений. Формулы интегрирования СОДУ могут входить в ММ независимо от компонентных уравнений, как это имеет место в (3.15), или быть интегрированными в ММ компонентов, как это выполнено в узловом методе.

От выбора метода решения СОДУ существенно зависят такие характеристики анализа, как точность и вычислительная эффективность. Эти характеристики определяются прежде всего типом и порядком выбранного метода интегрирования СОДУ.

Применяют два типа методов интегрирования — явные (иначе экстраполяционные или методы, основанные на формулах интегрирования вперед), и неявные (интерполяционные, основанные на формулах интегрирования назад). Различия между ними удобно показать на примере простейших методов первого порядка — методов Эйлера.

Формула *явного метода Эйлера* представляет собой следующую формулу замены производных в точке t_n:

$$d\mathbf{V}/dt\,|_{\,n} = (\mathbf{V}_{n+1} — \mathbf{V}_n) / h_n,$$

где индекс равен номеру шага интегрирования; $h_n = t_{n+1} - t_n$ — размер шага интегрирования (обычно h_n называют просто шагом интегрирования). В формуле *неявного метода Эйлера* использовано диф-

ференцирование назад:

$$d\mathbf{V}/dt\,|_n = (\mathbf{V}_n - \mathbf{V}_{n-1})\,/\,h_n,$$

где $h_n = t_n - t_{n-1}$.

Выполним сравнительный анализ явных и неявных методов на примере модельной задачи:

$$d\mathbf{V}/dt = \mathbf{AV} \qquad (3.23)$$

при ненулевых начальных условиях $\mathbf{V}_0 \neq 0$ и при использовании методов Эйлера с постоянным шагом h. Здесь **A** — постоянная матрица; **V** — вектор фазовых переменных.

При алгебраизации явным методом имеем

$$(\mathbf{V}_{n+1} - \mathbf{V}_n)\,/\,h = \mathbf{A}\,\mathbf{V}_n$$

или

$$\mathbf{V}_{n+1} = (\mathbf{E} + h\mathbf{A})\,\mathbf{V}_n,$$

где **E** — единичная матрица. Вектор $\mathbf{V}_{n+1}$ можно выразить через вектор начальных условий $\mathbf{V}_0$:

$$\mathbf{V}_{n+1} = (\mathbf{E} + h\mathbf{A})^n\,\mathbf{V}_0. \qquad (3.24)$$

Обозначим

$$\mathbf{B} = \mathbf{E} + h\mathbf{A} \qquad (3.25)$$

и применим преобразование подобия для матрицы **B**

$$\mathbf{B} = \mathbf{T}^{-1}\mathbf{diag}\{\lambda_{\mathbf{B}j}\}\mathbf{T},$$

где **T** — преобразующая матрица, $\mathbf{diag}\{\lambda_{\mathbf{B}j}\}$ -диагональная матрица с собственными значениями $\lambda_{\mathbf{B}j}$ матрицы **B** на диагонали. Нетрудно видеть, что

$$\mathbf{B}^n = \mathbf{T}^{-1}\mathbf{diag}\{\lambda_{\mathbf{B}j}^{\,n}\}\mathbf{T}.$$

Из линейной алгебры известно, что собственные значения матриц, связанных арифметическими операциями, оказываются связанными такими же преобразованиями. Поэтому из (3.25) следует

$$\lambda_{\mathbf{B}j} = 1 + h\lambda_{\mathbf{A}j}.$$

Точное решение модельной задачи (3.23) $\mathbf{V}(t) \to 0$ при $t \to \infty$, следовательно, условием устойчивости процесса численного решения можно считать

$$\mathbf{V}_{n+1} \to 0 \text{ при } n \to \infty,$$

откуда последовательно получаем

$$(\mathbf{E} + h\mathbf{A})^n\,\mathbf{V}_0 \to 0,$$

так как $\mathbf{V}_0 \neq 0$, то $(\mathbf{E} + h\mathbf{A})^n \to 0$, поскольку $\mathbf{T} \neq 0$, то $\lambda_{\mathbf{B}j}^{\,n} \to 0$ и условие устойчивости

$$-1 < |1 + h\lambda_{\mathbf{A}j}| < 1. \qquad (3.26)$$

Известно, что для физически устойчивых систем собственные значения матрицы коэффициентов в ММС оказываются отрицательными. Если к тому же все $\lambda_{\mathbf{A}j}$ вещественные величины (характер процессов в ММС с моделью (3.23) апериодический), то естественно определить *постоянные времени физической системы* как

$$\tau_j = -1\,/\,\lambda_{\mathbf{A}j},$$

и условие (3.26) конкретизируется следующим образом

$$-1 < |1 - h/\tau_j| < 1$$

или

$$0 < h < 2\tau_{min}, \qquad (3.27)$$

где τ_{min} — минимальная постоянная времени. Если использовать явные методы более высокого порядка, то может увеличиться коэффициент перед τ_{min} в (3.27), но это принципиально не меняет оценки явных методов.

Если нарушено условие (3.27), то происходит потеря устойчивости вычислений, а это означает, что в решении задачи возникают ложные колебания с увеличивающейся от шага к шагу амплитудой и быстрым аварийным остановом ЭВМ вследствие переполнения разрядной сетки. Конечно, ни о какой адекватности решения говорить не приходится.

Для соблюдения (3.27) применяют те или иные алгоритмы автоматического выбора шага. Отметим, что в сложной модели расчет τ_{min} для непосредственного выбора шага по (3.27) слишком трудоемок, кроме того, однократный расчет τ_{min} мало чем помогает, так как в нелинейных моделях τ_{min} может изменяться от шага к шагу.

Условие (3.27) накладывает жесткие ограничения на шаг интегрирования. В результате вычислительная эффективность явных методов резко падает с ухудшением *обусловленности* ММС. В самом деле, длительность $T_{инт}$ моделируемого процесса должна быть соизмеримой с временем успокоения системы после возбуждающего воздействия, т.е. соизмерима с максимальной постоянной времени τ_{max}. Требуемое число шагов интегрирования равно

$$Ш = T_{инт} / h \sim \tau_{max} / \tau_{min}.$$

Отношение $Ч = \tau_{max}/\tau_{min}$ называют *разбросом постоянных времени* или *числом обусловленности*. Чем больше это число, тем хуже обусловленность. Попытки применения явных методов к любым ММС чаще всего приводят к недопустимо низкой вычислительной эффективности, поскольку в реальных моделях $Ч > 10^5$ — обычная ситуация. Поэтому в настоящее время в универсальных программах анализа явные методы решения СОДУ не применяют.

Аналогичный анализ числовой устойчивости неявных методов дает следующие результаты. Вместо (3.24) имеем

$$\mathbf{V}_n = (\mathbf{E} - h\mathbf{A})^{-n}\, \mathbf{V}_0$$

и условие числовой устойчивости принимает вид

$$-1 < |1/(1 + h/\tau_j)| < 1,$$

которое выполняется при любых $h > 0$. Следовательно, неявный метод Эйлера обладает так называемой *A-устойчивостью*.

Примечание. Метод интегрирования СОДУ называют *A*-устойчивым, если погрешность интегрирования остается ограниченной при любом шаге $h > 0$.

Применение *A*-устойчивых методов позволяет существенно уменьшить требуемые числа шагов Ш. В этих методах шаг выбирается автоматически не из условий устойчивости, а только из соображений точности решения.

Выбор порядка метода решения СОДУ довольно прост: во-первых, более высокий порядок обеспечивает более высокую точность, во-вторых, среди неявных разностных методов, кроме метода Эйлера, *A*-устойчивы также методы второго порядка и среди них — метод трапеций. Поэтому преобладающее распространение в программах анализа получили методы второго порядка — модификации метода трапеций.

Алгоритм численного интегрирования СОДУ. Одна из удачных реализаций неявного метода второго порядка, которую можно считать модификацией *метода трапеций*, основана на комбинированном использовании явной и неявной формул Эйлера. Рассмотрим вопрос, почему такое комбинирование снижает погрешность и приводит к повышению порядка метода.

Предварительно отметим, что в методах *p*-го порядка локальная погрешность, т.е. погрешность, допущенная на одном *n*-м шаге интегрирования, оценивается старшим из отбрасываемых членов

$$\delta = c\|\mathbf{V}^{(p+1)}(\tau)\|\, h^{p+1},$$

в разложении решения $\mathbf{V}(t)$ в ряд Тейлора, где c — постоянный коэффициент, зависящий от метода, $\|\mathbf{V}^{(p+1)}(\tau)\|$ — норма вектора $(p+1)$-х производных $\mathbf{V}(t)$, которая оценивается с помощью конечно-разностной аппроксимации, τ — значение времени t внутри шага.

Если *n*-й шаг интегрирования в комбинированном методе был неявным, т.е. выполненным по неявной формуле, то следующий шаг с тем же значением *h* должен быть явным. Используя разложение

решения $\mathbf{V}(t)$ в ряд Тейлора в окрестностях точки t_{n+1}, получаем для (n+1)-го неявного шага

$$\mathbf{V}(t_n) = \mathbf{V}(t_{n+1}) - (d\mathbf{V}/dt)h_н + (d^2\mathbf{V}/dt^2)h_н^2 / 2! - (d^3\mathbf{V}/dt^3)h_н^3 / 3! + ..., \quad (3.28)$$

и для (n+2)-го явного шага

$$\mathbf{V}(t_{n+2}) = \mathbf{V}(t_{n+1}) + (d\mathbf{V}/dt)h_я + (d^2\mathbf{V}/dt^2)h_я^2/2! + (d^3\mathbf{V}/dt^3)h_я^3/3! + ..., \quad (3.29)$$

где $h_н$ *и* $h_я$ — величины неявного и явного шагов, а значения производных относятся к моменту t_{n+1}. Подставляя (3.28) в (3.29), при $h = h_я = h_н$ получаем:

$$\mathbf{V}(t_{n+2}) = \mathbf{V}(t_n) + 2(d\mathbf{V}/dt)h + 2(d^3\mathbf{V}/dt^3)h_я^3 / 3! + ...,$$

т.е. погрешности, обусловливаемые квадратичными членами в (3.28) и (3.29) взаимно компенсируются, и старшим из отбрасываемых членов становится член с h^3. Следовательно, изложенное комбинирование неявной и явной формул Эйлера дает метод интегрирования второго порядка.

Неявные методы и, в частности, рассмотренный комбинированный метод целесообразно использовать только при переменной величине шага. Действительно, при заметных скоростях изменения фазовых переменных погрешность остается в допустимых пределах только при малых шагах, в квазистатических режимах шаг может быть во много раз больше.

Алгоритмы автоматического выбора шага основаны на сравнении допущенной и допустимой локальных погрешностей. Например, вводится некоторый диапазон (коридор) погрешностей δ, в пределах которого шаг сохраняется неизменным. Если же допущенная погрешность превышает верхнюю границу диапазона, то шаг уменьшается, если же выходит за нижнюю границу, то шаг увеличивается.

Методы решения систем нелинейных алгебраических уравнений. Вычисления при решении СОДУ состоят из нескольких вложенных один в другой циклических процессов. Внешний цикл — цикл пошагового численного интегрирования, параметром цикла является номер шага. Если модель анализируемого объекта нелинейна, то на каждом шаге выполняется промежуточный цикл — итерационный цикл решения системы нелинейных алгебраических уравнений (СНАУ). Параметр цикла — номер итерации. Во внутреннем цикле решается система линейных алгебраических уравнений (СЛАУ), например, при применении узлового метода формирования ММС такой системой является (3.19). Поэтому в математическое обеспечение анализа на макроуровне входят методы решения СНАУ и СЛАУ.

Для решения СНАУ можно применять прямые итерационные методы такие, как метод простой итерации или метод Зейделя, но в современных программах анализа наибольшее распространение получил метод Ньютона, основанный на линеаризации СНАУ. Собственно модель (3.19) получена именно в соответствии с методом Ньютона. Основное преимущество метода Ньютона — высокая скорость сходимости.

Представим СНАУ в виде

$$\mathbf{F}(\mathbf{X}) = 0. \quad (3.30)$$

Разлагая $\mathbf{F}(\mathbf{X})$ в ряд Тейлора в окрестностях некоторой точки $\mathbf{X}_k$, получаем

$$\mathbf{F}(\mathbf{X}) = \mathbf{F}(\mathbf{X}_k) + (\partial\mathbf{F}/\partial\mathbf{X})(\mathbf{X}-\mathbf{X}_k) + (\mathbf{X}-\mathbf{X}_k)^T(\partial^2\mathbf{F}/\partial\mathbf{X}^2)(\mathbf{X}-\mathbf{X}_k) / 2 + ... = 0.$$

Сохраняя только линейные члены, получаем СЛАУ с неизвестным вектором $\mathbf{X}$:

$$\mathbf{Я}_k(\mathbf{X} - \mathbf{X}_k) = - \mathbf{F}(\mathbf{X}_k), \quad (3.31)$$

где $\mathbf{Я}_k = (\partial\mathbf{F}/\partial\mathbf{X})|_k$. Решение системы (3.31) дает очередное приближение к корню системы (3.30), которое удобно обозначить $\mathbf{X}_{k+1}$.

Вычислительный процесс стартует с начального приближения $\mathbf{X}_0$ и в случае сходимости итераций заканчивается, когда погрешность, оцениваемая как

$$|\Delta\mathbf{X}_k| = |\mathbf{X}_k - \mathbf{X}_{k-1}|,$$

станет меньше допустимой погрешности ε.

Однако метод Ньютона не всегда приводит к сходящимся итерациям. Условия сходимости метода Ньютона выражаются довольно сложно, но существует легко используемый подход к улучшению схо-

димости. Это близость начального приближения к искомому корню СНАУ. Использование этого фактора привело к появлению метода решения СНАУ, называемого *продолжением решения по параметру*.

В методе продолжения решения по параметру в ММС выделяется некоторый параметр α, такой, что при $\alpha = 0$ корень $\mathbf{X}_{\alpha=0}$ системы (3.30) известен, а при увеличении α от 0 до его истинного значения составляющие вектора **X** плавно изменяются от $\mathbf{X}_{\alpha=0}$ до истинного значения корня. Тогда задача разбивается на ряд подзадач, последовательно решаемых при меняющихся значениях α, и при достаточно малом шаге $\Delta\alpha$ изменения α условия сходимости выполняются.

В качестве параметра α можно выбрать некоторый внешний параметр, например, при анализе электронных схем им может быть напряжение источника питания. Но на практике при интегрировании СОДУ в качестве α выбирают шаг интегрирования h. Очевидно, что при $h = 0$ корень СНАУ равен значению вектора неизвестных на предыдущем шаге. Регулирование значений h возлагается на алгоритм автоматического выбора шага.

В этих условиях очевидна целесообразность представления математических моделей для анализа статических состояний в виде СОДУ, как и для анализа динамических режимов.

Методы решения систем линейных алгебраических уравнений. В программах анализа в САПР для решения СЛАУ чаще всего применяют метод Гаусса или его разновидности. Метод Гаусса — метод последовательного исключения неизвестных из системы уравнений. При исключении k-й неизвестной x_k из системы уравнений

$$\mathbf{AX} = \mathbf{B} \qquad (3.32)$$

все коэффициенты a_{ij} при $i>k$ и $j>k$ пересчитывают по формуле

$$a_{ij} = a_{ij} - a_{ik}\, a_{kj} / a_{kk}. \qquad (3.33)$$

Исключение n-1 неизвестных, где n — порядок системы (3.32), называют прямым ходом, в процессе которого матрица коэффициентов приобретает треугольный вид. При обратном ходе последовательно вычисляют неизвестные, начиная с x_n.

В общем случае число арифметических операций для решения (3.32) по Гауссу пропорционально n^3. Это приводит к значительным затратам машинного времени, поскольку СЛАУ решается многократно в процессе одновариантного анализа, и существенно ограничивает сложность анализируемых объектов.

Заметно повысить вычислительную эффективность анализа можно, если использовать характерное практически для всех приложений свойство высокой разреженности матрицы **A** в модели (3.32).

Матрицу называют *разреженной*, если большинство ее элементов равно нулю. Эффективность обработки разреженных матриц велика потому, что, во-первых, пересчет по формуле (3.33) не требуется, если хотя бы один из элементов a_{ik} или a_{kj} оказывается нулевым, во-вторых, не требуются затраты памяти для хранения нулевых элементов. Хотя алгоритмы обработки разреженных матриц более сложны, но в результате удается получить затраты машинного времени, близкие к линейным, например, затраты оказываются пропорциональными $n^{1,2}$.

При использовании методов разреженных матриц нужно учитывать зависимость вычислительной эффективности от того, как представлена матрица коэффициентов **A,** точнее от того, в каком порядке записаны ее строки и столбцы.

Для пояснения этой зависимости рассмотрим два варианта представления одной и той же СЛАУ. В первом случае система уравнений имеет вид

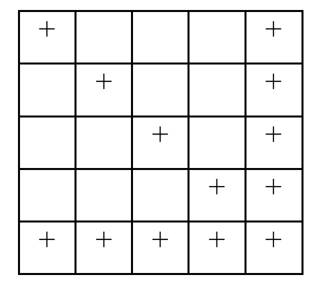
$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 + a_{15}x_5 = b_1;$$
$$a_{21}x_1 + a_{22}x_2 = b_2;$$
$$a_{31}x_1 + a_{33}x_3 = b_3;$$
$$a_{41}x_1 + a_{44}x_4 = b_4;$$
$$a_{51}x_1 + a_{55}x_5 = b_5.$$

При прямом ходе в соответствии с формулой (3.33) все элементы матрицы, которые первоначально были нулевыми, становятся ненулевыми, а матрица оказывается полностью *насыщенной*. Эле-

Таблица 3.3

+	+	+	+	+
+	+	.	.	.
+	.	+	.	.
+	.	.	+	.
+	.	.	.	+

Таблица 3.4



+				+
	+			+
		+		+
			+	+
+	+	+	+	+

менты, становящиеся ненулевыми в процессе гауссовых исключений, называют *вторичными ненулями*. Вторичные ненули в таблице 3.3 отмечены точкой.

Во втором случае меняются местами первое и пятое уравнения. Матрицы коэффициентов имеют вид таблиц 3.3 и 3.4, где ненулевые элементы представлены знаком +. Теперь вторичные ненули не появляются, матрица остается разреженной, высокая вычислительная эффективность сохраняется.

Таким образом, методы разреженных матриц должны включать в себя способы *оптимального упорядочения строк и столбцов* матриц. Используют несколько критериев оптимальности упорядочения. Простейшим из них является критерий расположения строк в порядке увеличения числа первичных ненулей, более сложные критерии учитывают не только первичные ненули, но и появляющиеся вторичные ненули.

Методом разреженных матриц называют метод решения СЛАУ на основе метода Гаусса с учетом разреженности (первичной и вторичной) матрицы коэффициентов.

Метод разреженных матриц можно реализовать путем интерпретации и компиляции. В обоих случаях создаются массивы ненулевых коэффициентов матрицы (с учетом вторичных ненулей) и массивы координат этих ненулевых элементов.

При этом выигрыш в затратах памяти довольно значителен. Так, при матрице умеренного размера 200×200 без учета разреженности потребуется 320 кбайт. Если же взять характерное значение 9 для среднего числа ненулей в одной строке, то для коэффициентов и указателей координат потребуется не более 28 кбайт.

В случае *интерпретации* моделирующая программа для каждой операции по (3.33) при $a_{ik} \neq 0$ и $a_{kj} \neq 0$ находит, используя указатели, нужные коэффициенты и выполняет арифметические операции по (3.33). Поскольку СЛАУ в процессе анализа решается многократно, то и операции поиска нужных коэффициентов также повторяются многократно, на что естественно тратится машинное время.

Способ *компиляции* более экономичен по затратам времени, но уступает способу интерпретации по затратам памяти. При компиляции поиск нужных для (3.33) коэффициентов выполняется однократно перед численным решением задачи. Вместо непосредственного выполнения арифметических операций для каждой из них компилируется команда с найденными адресами ненулевых коэффициентов. Такие команды образуют рабочую программу решения СЛАУ, которая и будет решаться многократно. Очевидно, что теперь в рабочей программе будет выполняться минимально необходимое число арифметических операций.

Анализ в частотной области. Анализ в частотной области выполняется по отношению к линеаризованным моделям объектов. Для линейных СОДУ справедливо применение для алгебраизации дифференциальных уравнений преобразования Фурье, в котором оператор d/dt заменяется на оператор $j\omega$.

Характерной особенностью получающейся СЛАУ является комплексный характер матрицы коэффициентов, что в некоторой степени усложняет процедуру решения, но не создает принципиальных трудностей. При решении задают ряд частот ω_k. Для каждой частоты решают СЛАУ и определяют действительные и мнимые части искомых фазовых переменных. По ним определяют амплитуду и фазовый угол каждой спектральной составляющей, что и позволяет построить амплитудно-частотные, фазочастотные характеристики, найти собственные частоты колебательной системы и т.п.

Многовариантный анализ. Одновариантный анализ позволяет получить информацию о состоянии и поведении проектируемого объекта в одной точке пространства внутренних **X** и внешних **Q** параметров. Очевидно, что для оценки свойств проектируемого объекта этого недостаточно. Нужно выполнять *многовариантный анализ*, т.е. исследовать поведение объекта, в ряде точек упомянутого пространства, которое для краткости будем далее называть *пространством аргументов*.

Чаще всего многовариантный анализ в САПР выполняется в интерактивном режиме, когда разработчик неоднократно меняет в математической модели те или иные параметры из множеств **X** и **Q**, выполняет одновариантный анализ и фиксирует полученные значения выходных параметров. Подоб-

ный многовариантный анализ позволяет оценить *области работоспособности*, степень выполнения условий работоспособности, а следовательно, степень выполнения технического задания (ТЗ) на проектирование, разумность принимаемых промежуточных решений по изменению проекта и т.п.

Примечание. Областью работоспособности называют область в пространстве аргументов, в пределах которой выполняются все заданные условия работоспособности, т.е. значения всех выходных параметров находятся в допустимых по ТЗ пределах.

Как упомянуто в гл. 1, среди процедур многовариантного анализа можно выделить типовые, выполняемые по заранее составленным программам. К таким процедурам относятся анализ чувствительности и статистический анализ.

Наиболее просто *анализ чувствительности* реализуется путем численного дифференцирования. Пусть анализ проводится в некоторой точке $\mathbf{X}_{ном}$ пространства аргументов, в которой предварительно проведен одновариантный анализ и найдены значения выходных параметров $y_{jном}$. Выделяется N параметров-аргументов x_i (из числа элементов векторов **X** и **Q**), влияние которых на выходные параметры подлежит оценить, поочередно каждый из них получает приращение Δx_i, выполняется одновариантный анализ, фиксируются значения выходных параметров y_j и подсчитываются значения абсолютных

$$A_{ji} = (y_j - y_{j\ ном}) / \Delta x_i$$

и относительных коэффициентов чувствительности

$$B_{ji} = A_{ji}\, x_{iном} / y_{jном}.$$

Такой метод численного дифференцирования называют *методом приращений*. Для анализа чувствительности, согласно методу приращений, требуется выполнить N+1 раз одновариантный анализ. Результат его применения — матрицы абсолютной и относительной чувствительности, элементами которых являются коэффициенты A_{ji} и B_{ji}.

Примечание. Анализ чувствительности – это расчет векторов градиентов выходных параметров, который входит составной частью в программы параметрической оптимизации, использующие градиентные методы.

Цель с*татистического анализа* — оценка законов распределения выходных параметров и (или) числовых характеристик этих распределений. Случайный характер величин y_j обусловлен случайным характером параметров элементов x_i, поэтому исходными данными для статистического анализа являются сведения о законах распределения x_i. В соответствии с результатами статистического анализа прогнозируют такой важный производственный показатель, как процент бракованных изделий в готовой продукции (рис. 3.8). На рисунке представлена рассчитанная плотность P распределения выходного параметра y_j, имеющего условие работоспособности $y_j<T_j$, затемненный участок характеризует долю изделий, не удовлетворяющих условию работоспособности параметра y_j.

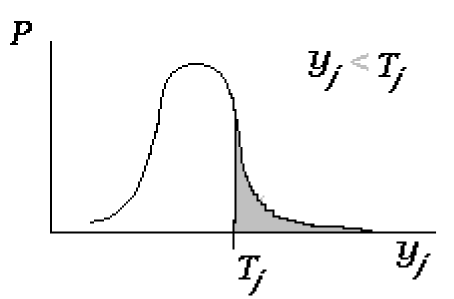


Рис. 3.8. Иллюстрация определения процента выпуска негодных изделий

В САПР статистический анализ осуществляется численным методом — *методом Монте-Карло* (статистических испытаний). В соответствии с этим методом выполняются N статистических испытаний, каждое статистическое испытание представляет собой одновариантный анализ, выполняемый при случайных значениях параметров-аргументов. Эти случайные значения выбирают в соответствии с заданными законами распределения аргументов x_i. Полученные в каждом испытании значения выходных параметров накапливают, после N испытаний обрабатывают, что дает следующие результаты:

— гистограммы выходных параметров;

— оценки математических ожиданий и дисперсий выходных параметров:

— оценки коэффициентов корреляции и регрессии между избранными выходными и внутренними параметрами, которые, в частности, можно использовать для оценки коэффициентов чувствительности.

Статистический анализ, выполняемый в соответствии с методом Монте-Карло, — трудоемкая процедура, поскольку число испытаний N приходится выбирать довольно большим, чтобы достичь приемлемой точности анализа. Другая причина, затрудняющая применение метода Монте-Карло, —

трудности в получении достоверной исходной информации о законах распределения параметров-аргументов x_i.

Более типична ситуация, когда законы распределения x_i неизвестны, но с большой долей уверенности можно указать предельно допустимые отклонения Δx_i параметров x_i от номинальных значения $x_{i\text{ном}}$ (такие отклонения часто указываются в паспортных данных на комплектующие детали). В таких случаях более реалистично применять *метод анализа на наихудший случай*. Согласно этому методу, сначала выполняют анализ чувствительности с целью определения знаков коэффициентов чувствительности. Далее осуществляют m раз одновариантный анализ, где m -число выходных параметров. В каждом варианте задают значения аргументов, наиболее неблагоприятные для выполнения условия работоспособности очередного выходного параметра y_j, $j \in [1{:}m]$. Так, если $y_j<T_j$ и коэффициент чувствительности положительный (т.е. sign(B_{ji}) = 0) или $y_j>T_j$ и sign(B_{ji}) = 1, то

$$x_i = x_{i\text{ном}} + \Delta x_i,$$

иначе

$$x_i = x_{i\text{ном}} - \Delta x_i.$$

Однако следует заметить, что, проводя анализ на наихудший случай, можно получить завышенные значения разброса выходных параметров, и если добиваться выполнения условий работоспособности в наихудших случаях, то это часто ведет к неоправданному увеличению стоимости, габаритных размеров, массы и других показателей проектируемых конструкций, хотя и гарантирует с запасом выполнение условий работоспособности.

Организация вычислительного процесса в универсальных программах анализа на макроуровне. На рис. 3.9 представлена граф-схема вычислительного процесса при анализе во временной области на макроуровне. Алгоритм отражает решение системы алгебро-дифференциальных уравнений

$$\boldsymbol{\varphi}(d\mathbf{V}/dt, \mathbf{V}, t) = 0.$$

На каждом шаге численного интегрирования решается система нелинейных алгебраических уравнений

$$\mathbf{F}(\mathbf{X}) = 0$$

методом Ньютона. На каждой итерации выполняется решение системы линейных алгебраических уравнений

$$\mathbf{Я}\Delta\mathbf{X} = \mathbf{B}.$$

Другие используемые обозначения:

$\mathbf{V}_0(t_0)$ — начальные условия;

h и $h_{\text{нач}}$ — шаг интегрирования и его начальное значение;

$\mathbf{U}_{\text{вн}}(t)$ — вектор внешних воздействий;

N и $N_{\text{д}}$ — число ньютоновских итераций и его максимально допустимое значение;

ε — предельно допустимая погрешность решения СНАУ;

δ — погрешность, допущенная на одном шаге интегрирования;

$m1$ — максимально допустимое значение погрешности интегрирования на одном шаге;

$m2$ — нижняя граница коридора рациональных погрешностей интегрирования.

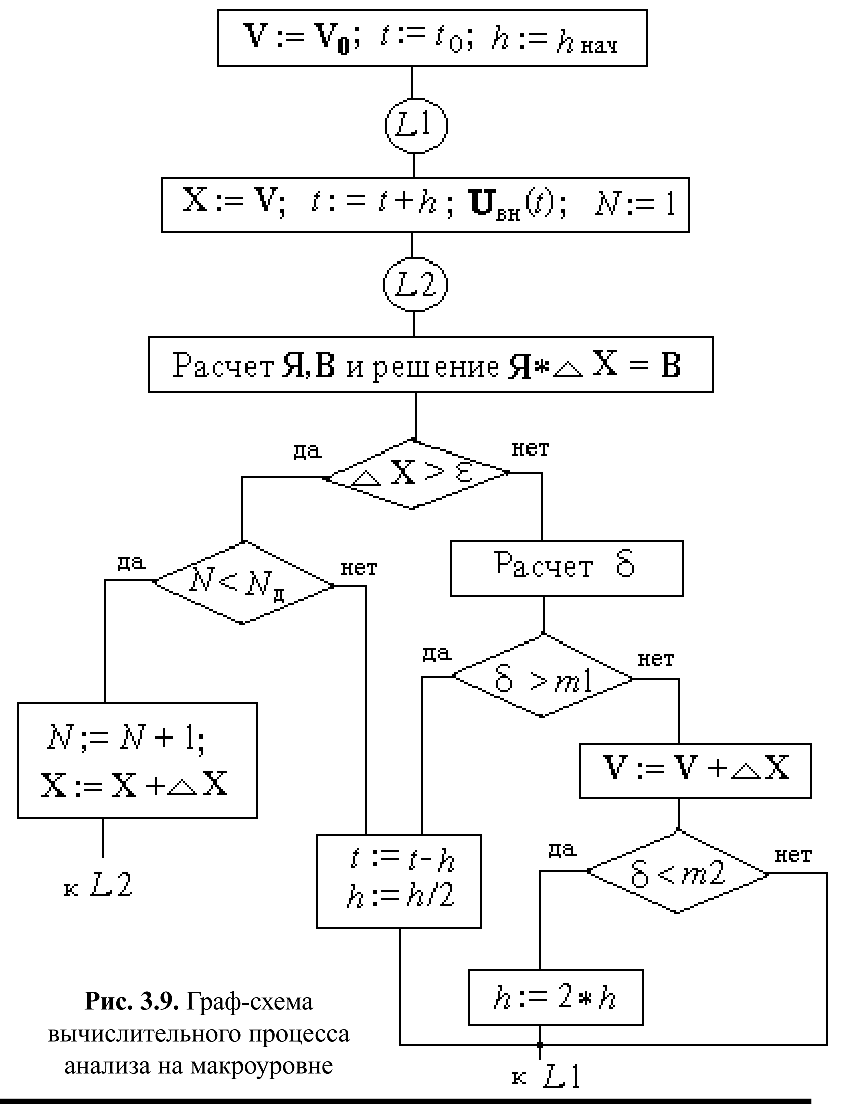


Рис. 3.9. Граф-схема вычислительного процесса анализа на макроуровне

Из рисунка ясно, что при $N \geq N_д$ фиксируется несходимость ньютоновских итераций и после дробления шага происходит возврат к интегрированию при тех же начальных для данного шага условиях. При сходимости рассчитывается δ и в зависимости от того, выходит погрешность за пределы диапазона [$m2$, $m1$] или нет шаг изменяется либо сохраняет свое прежнее значение.

Параметры $N_д$, $m1$, $m2$, ε, $h_{нач}$ задаются “по умолчанию” и могут настраиваться пользователем.

Матрицу Якоби **Я** и вектор правых частей **B** необходимо рассчитывать по программе, составляемой для каждого нового исследуемого объекта. Составление программы выполняет компилятор, входящий в состав программного комплекса анализа. Общая структура такого комплекса представлена на рис. 3.10.

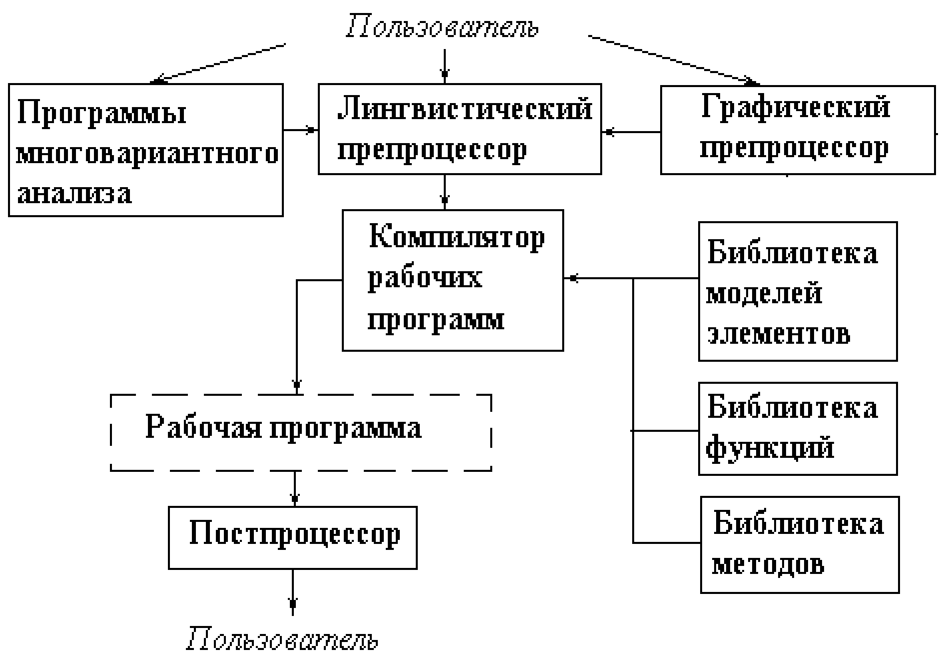


Рис. 3.10. Структура программного комплекса анализа на макроуровне

Исходные данные об объекте можно задавать в графическом виде (в виде эквивалентной схемы) или на входном языке программы анализа. Запись на таком языке обычно представляет собой список компонентов анализируемого объекта с указанием их взаимосвязей. Вводимые данные преобразуются во внутреннее представление с помощью графического и лингвистического препроцессоров, в которых предусмотрена также диагностика нарушений формальных языковых правил. Графическое представление более удобно, особенно для малоопытных пользователей.

Задав описание объекта, пользователь может приступить к многовариантному анализу либо по одной из программ такого анализа, либо в интерактивном режиме, изменяя условия моделирования между вариантами с помощью лингвистического препроцессора.

Наиболее сложная часть комплекса — компилятор рабочих программ, именно в нем создаются программы расчета матрицы Якоби **Я** и вектора правых частей **B**, фигурирующих в вычислительном процессе (см. рис. 3.9). Собственно рабочая программа (см. рис. 3.10) — это и есть программа процесса, показанного на рис. 3.9. Для каждого нового моделируемого объекта составляются свои рабочие программы. При компиляции используются заранее разработанные математические модели типовых компонентов, известные функции для отображения входных воздействий и т.п. из соответствующих библиотек.

Постпроцессор представляет результаты анализа в табличной и графической формах, это могут быть зависимости фазовых переменных от времени, значения выходных параметров-функционалов и т.п.

3.4. Математическое обеспечение анализа на микроуровне

Математические модели на микроуровне. Математическими моделями на микроуровне являются дифференциальные уравнения в частных производных или интегральные уравнения, описывающие поля физических величин. Другими словами, на микроуровне используются модели с распре-

деленными параметрами. В качестве независимых переменных в моделях могут фигурировать пространственные переменные x_1, x_2, x_3 и время t.

Характерными примерами моделей могут служить уравнения математической физики вместе с заданными краевыми условиями.

Например:

1) уравнение теплопроводности

$$C \rho \, \partial T / \partial t = \mathrm{div} \, (\lambda \, \mathbf{grad} \, T) + g,$$

где C — удельная теплоемкость, ρ — плотность, T — температура, t — время, λ — коэффициент теплопроводности, g — количество теплоты, выделяемой в единицу времени в единице объема;

2) уравнение диффузии

$$\partial N / \partial t = \mathrm{div} \, (D \, \mathbf{grad} \, N) \,,$$

где N — концентрация частиц, D — коэффициент диффузии;

3) уравнения непрерывности, используемые в физике полупроводниковых приборов:

для дырок

$$\partial p / \partial t = - (1/q) \, \mathrm{div} \, \mathbf{J}_p + g_p \,,$$

для электронов

$$\partial n / \partial t = (1/q) \, \mathrm{div} \, \mathbf{J}_\mathrm{n} + g_n \,,$$

и уравнение Пуассона

$$\mathrm{div} \, \mathbf{E} = \rho / (\varepsilon \, \varepsilon_0),$$

Здесь p и n — концентрации дырок и электронов; q — заряд электрона; $\mathbf{J}_p$ и $\mathbf{J}_n$ — плотности дырочного и электронного токов; g_p и g_n — скорости процессов генерации-рекомбинации дырок и электронов; $\mathbf{E}$ — напряженность электрического поля;, ρ — плотность электрического заряда; ε и ε_0 — диэлектрическая проницаемость и диэлектрическая постоянная.

Краевые условия включают начальные условия, характеризующие пространственное распределение зависимых переменных в начальный момент времени, и граничные, задающие значения этих переменных на границах рассматриваемой области в функции времени.

Методы анализа на микроуровне. В САПР решение дифференциальных или интегро-дифференциальных уравнений с частными производными выполняется численными методами. Эти методы основаны на дискретизации независимых переменных — их представлении конечным множеством значений в выбранных узловых точках исследуемого пространства. Эти точки рассматриваются как узлы некоторой сетки, поэтому используемые в САПР методы — это *сеточные* методы.

Среди сеточных методов наибольшее распространение получили два метода: метод конечных разностей (МКР) и метод конечных элементов (МКЭ). Обычно выполняют дискретизацию пространственных независимых переменных, т.е. используют пространственную сетку. В этом случае результатом дискретизации является система обыкновенных дифференциальных уравнений для задачи нестационарной или система алгебраических уравнений для стационарной.

Пусть необходимо решить уравнение

$$LV(\mathbf{z}) = f(\mathbf{z})$$

с заданными краевыми условиями

$$MV(\mathbf{z}) = \psi(\mathbf{z}),$$

где L и M — дифференциальные операторы, $V(\mathbf{z})$ — фазовая переменная, $\mathbf{z} = (x_1, x_2, x_3, t)$ — вектор независимых переменных, $f(\mathbf{z})$ и $\psi(\mathbf{z})$ — заданные функции независимых переменных.

В *методе конечных разностей* алгебраизация производных по пространственным координатам базируется на аппроксимации производных конечно-разностными выражениями. При использовании метода нужно выбрать шаги сетки по каждой координате и вид шаблона. Под шаблоном понимают множество узловых точек, значения переменных в которых используются для аппроксимации производной в одной конкретной точке.

Примеры шаблонов для одномерных и двумерных задач приведены на рис. 3.11. На этом рисунке кружком большего диаметра обозначены узлы, в которых аппроксимируется производная. Черными точками обозначены узлы, значения фазовой переменной в которых входят в аппроксимирующее выражение. Число, записанное около узла, равно коэффициенту, с которым значение фазовой переменной входит в аппроксимирующее выражение. Так, для одномерных шаблонов в

верхней части рисунка показана аппроксимация производной $\partial V/\partial x$ в точке k, и указанным шаблонам при их просмотре слева направо соответствуют аппроксимации

$$h(\partial V/\partial x) = V_{k+1} - V_k;\ 2h(\partial V/\partial x) = V_{k+1} - V_{k-1};\ h^2(\partial^2 V/\partial x^2) = V_{k+1} - 2V_k + V_{k-1},$$

где h — шаг дискретизации по оси x.

Шаблоны для двумерных задач в нижней части рис. 3.11 соответствуют следующим конечно-разностным операторам: левый рисунок -

$$h^2\nabla^2 V = C^2(\partial^2 V/\partial x_1^2 + \partial^2 V/\partial x_2^2) = V_{k+1,j} + V_{k-1,j} + V_{k,j+1} + V_{k,j-1} - 4V_{k,j},$$

средний рисунок -

$$2h^2\nabla^2 V = V_{k+1,j+1} + V_{k-1,j+1} + V_{k+1,j-1} + V_{k-1,j-1} - 4V_{k,j},$$

правый рисунок -

$$4h^2\partial^2 V/\partial x_1 \partial x_2 = V_{k+1,j+1} - V_{k-1,j+1} - V_{k+1,j-1} + V_{k-1,j-1}.$$

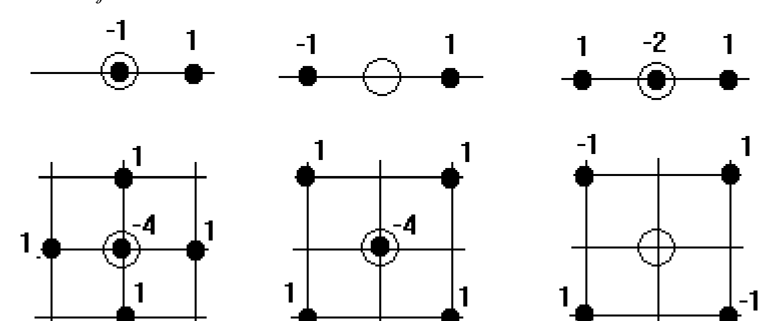


Рис. 3.11. Примеры шаблонов для метода конечных разностей

Здесь $V_{k,j}$ — значение V в точке (x_{1k}, x_{2j}); приняты одинаковые значения шагов h по обеим координатам.

Метод конечных элементов основан на аппроксимации не производных, а самого решения $V(\mathbf{z})$. Но поскольку оно неизвестно, то аппроксимация выполняется выражениями с неопределенными коэффициентами q_i

$$U(\mathbf{z}) = \mathbf{Q}^T \boldsymbol{\varphi}(\mathbf{z}), \qquad (3.34)$$

где $\mathbf{Q}^T = (q_1, q_2,...q_n)^T$- вектор-строка неопределенных коэффициентов, $\boldsymbol{\varphi}(\mathbf{z})$ — вектор-столбец *координатных* (иначе опорных) функций, заданных так, что удовлетворяются граничные условия.

При этом речь идет об аппроксимациях решения в пределах конечных элементов, а с учетом их малых размеров можно говорить об использовании сравнительно простых аппроксимирующих выражений $U(\mathbf{z})$ (например, $\boldsymbol{\varphi}(\mathbf{z})$ — полиномы низких степеней). В результате подстановки $U(\mathbf{z})$ в исходное дифференциальное уравнение и выполнения операций дифференцирования получаем систему невязок

$$\Delta(\mathbf{z}, \mathbf{Q}) = LU(\mathbf{z}) - f(\mathbf{z}) = L(\mathbf{Q}^T\boldsymbol{\varphi}(\mathbf{z})) - f(\mathbf{z}), \qquad (3.35)$$

из которой требуется найти вектор **Q**.

Эту задачу (определение **Q)** решают одним из следующих методов:

1) *метод коллокаций*, в котором, используя (3.35), формируют n уравнений с неизвестным вектором **Q:**

$$L(\mathbf{Q}^T\boldsymbol{\varphi}(\mathbf{z}_i)) - f(\mathbf{z}_i) = 0, \quad i = 1, 2,...n,$$

где n — число неопределенных коэффициентов;

2) *метод наименьших квадратов*, основанный на минимизации квадратов невязок (3.35) в n точках или в среднем по рассматриваемой области;

3) *метод Галеркина*, с помощью которого минимизируются в среднем по области невязки со специально задаваемыми весовыми коэффициентами.

Наибольшее распространение МКЭ получил в САПР машиностроения для анализа прочности объектов. Для этой задачи можно использовать рассмотренный подход, т.е. выполнить алгебраизацию исходного уравнения упругости (уравнения Ламе). Однако более удобным в реализации МКЭ оказался подход, основанный на вариационных принципах механики.

МКЭ в программах анализа механической прочности. В качестве исходного положения принимают вариационный принцип Лагранжа (принцип потенциальной энергии), в соответствии с которым равновесное состояние, в которое может прийти система, характеризуется минимумом потенциальной энергии.

Потенциальная энергия П определяется как разность энергии Э деформации тела и работы А массовых и приложенных поверхностных сил.

В свою очередь

$$Э = 0{,}5 \int_{\mathbf{R}} \boldsymbol{\varepsilon}^{\mathrm{T}} \boldsymbol{\sigma}\, d\mathbf{R}, \qquad (3.36)$$

где $\boldsymbol{\varepsilon}^{\mathrm{T}} = (\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{33}, \varepsilon_{12}, \varepsilon_{13}, \varepsilon_{23})^{\mathrm{T}}$ — вектор-строка деформаций, $\boldsymbol{\sigma} = (\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{13}, \sigma_{23})$ — вектор-столбец напряжений, **R** — рассматриваемая область. Деформации ε_{ij} можно выразить через перемещения

$$\varepsilon_{ij} = 0{,}5(\partial W_i/\partial x_j + \partial W_j/\partial x_i), \qquad (3.37)$$

где W_i — перемещение вдоль оси x_i, или в матричной форме

$$\boldsymbol{\varepsilon} = 0{,}5\ \mathbf{SW}, \qquad (3.38)$$

где **S** — очевидный из (3.37) оператор дифференцирования.

Деформации и напряжения связаны между собой с помощью матрицы **D**, характеризующей упругие свойства среды, которая представлена в табл. 3.5:

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon}. \qquad (3.39)$$

Коэффициенты λ и μ, фигурирующие в таблице, называют постоянными Ламе, они выражают упругие свойства материала детали.

Таблица 3.5

λ+2μ	λ	0	0	0	0
λ	λ+2μ	λ	0	0	0
0	λ	λ+2μ	0	0	0
0	0	0	2μ	0	0
0	0	0	0	2μ	0
0	0	0	0	0	2μ

Подставляя (3.39) и (3.38) в (3.36), получаем

$$Э = 0{,}5 \int_{\mathbf{R}} \mathbf{W}^{\mathrm{T}} \mathbf{S}^{\mathrm{T}} \mathbf{DSW}\, d\mathbf{R},$$

Решением задачи должно быть поле перемещений **W(X)**, где $\mathbf{X} = (x_1, x_2, x_3)$. В соответствии с МКЭ это решение аппроксимируется с помощью функций (3.34), которые применительно к совокупности конечных элементов представим в матричной форме:

$$\mathbf{U}(\mathbf{X}) = \mathbf{NQ},$$

где **N** — матрица координатных функций, **Q** — вектор неопределенных коэффициентов. Заменяя **W**(**X**) на **U**(**X**), получаем

$$Э = 0{,}5 \int_{\mathbf{R}} \mathbf{Q}^{\mathrm{T}} \mathbf{N}^{\mathrm{T}} \mathbf{S}^{\mathrm{T}} \mathbf{DSN\, Q}\, d\mathbf{R} = 0{,}5\ \mathbf{Q}^{\mathrm{T}}\Big(\int_{\mathbf{R}} (\mathbf{SN})^{\mathrm{T}} \mathbf{DSN}\, d\mathbf{R}\Big)\, \mathbf{Q} = 0{,}5\ \mathbf{Q}^{\mathrm{T}} \mathbf{K\, Q}, \qquad (3.40)$$

где $\mathbf{K} = \int_{\mathbf{R}} (\mathbf{SN})^{\mathrm{T}} \mathbf{DSN}\, d\mathbf{R}$ — матрица жесткости.

В соответствии с принципом потенциальной энергии в состоянии равновесия имеем

$$\partial П/\partial \mathbf{Q} = \partial Э/\partial \mathbf{Q} - \partial A/\partial \mathbf{Q} = 0$$

или, дифференцируя (3.40), находим

$$\mathbf{KQ} = \mathbf{B}, \qquad (3.41)$$

где $\mathbf{B} = \partial A/\partial \mathbf{Q}$ — вектор нагрузок. Таким образом, задача анализа прочности, согласно МКЭ, сведена к решению системы линейных алгебраических уравнений (3.41).

Матрица жесткости также оказывается сильно разреженной, поэтому для решения (3.41) применяют методы разреженных матриц.

Примечание. Одним из широко известных методов разреженных матриц является метод прогонки, применяемый в случае трехдиагональных матриц коэффициентов в системе алгебраических уравнений.

Структура программ анализа по МКЭ на микроуровне. Основными частями программы анализа по МКЭ являются библиотеки конечных элементов, препроцессор, решатель и постпроцессор.

Библиотеки конечных элементов (КЭ) содержат модели КЭ — их матрицы жесткости. Очевидно, что модели КЭ будут различными для разных задач (анализ упругих или пластических деформаций, моделирование полей температур, электрических потенциалов и т.п.), разных форм КЭ (например, в двумерном случае — треугольные или четырехугольные элементы), разных наборов координатных функций.

Исходные данные для *препроцессора* — геометрическая модель объекта, чаще всего получаемая

из подсистемы конструирования. Основная функция препроцессора — представление исследуемой среды (детали) в сеточном виде, т.е. в виде множества конечных элементов.

Решатель — программа, которая ассемблирует (собирает) модели отдельных КЭ в общую систему алгебраических уравнений (3.41) и решает эту систему одним из методов разреженных матриц.

Постпроцессор служит для визуализации результатов решения в удобной для пользователя форме. В машиностроительных САПР это графическая форма. Пользователь может видеть исходную (до нагружения) и деформированную формы детали, поля напряжений, температур, потенциалов и т.п. в виде цветных изображений, в которых палитра цветов или интенсивность свечения характеризуют значения фазовой переменной.

Мировыми лидерами среди программ конечно-элементного анализа являются программно-методические комплексы Nastran, Ansys, Nisa, Adina, Cosmos.

Как правило, эти комплексы включают в себя ряд программ, родственных по математическому обеспечению, интерфейсам, общности некоторых используемых модулей. Эти программы различаются ориентацией на разные приложения, степенью специализации, ценой или выполняемой обслуживающей функцией. Например, в комплексе Ansys основные решающие модули позволяют выполнять анализ механической прочности, теплопроводности, динамики жидкостей и газов, акустических и электромагнитных полей. Во все варианты программ входят пре- и постпроцессоры, а также интерфейс с базой данных. Предусмотрен экспорт (импорт) данных между Ansys и ведущими комплексами геометрического моделирования и машинной графики.

3.5. Математическое обеспечение анализа на функционально-логическом уровне

Моделирование и анализ аналоговых устройств. На функционально-логическом уровне исследуют устройства, в качестве элементов которых принимают достаточно сложные узлы и блоки, считавшиеся системами на макроуровне. Поэтому необходимо упростить представление моделей этих узлов и блоков по сравнению с их представлением на макроуровне. Другими словами, вместо полных моделей узлов и блоков нужно использовать их макромодели.

Вместо двух типов фазовых переменных в моделях функционально-логического уровня фигурируют переменные одного типа, называемые *сигналами*. Физический смысл сигнала, т.е. его отнесение к фазовым переменным типа потока или типа потенциала, конкретизируют в каждом случае исходя из особенностей задачи.

Основой моделирования аналоговых устройств на функционально-логическом уровне является использование аппарата передаточных функций. При этом модель каждого элемента представляют в виде уравнения вход-выход, т.е. в виде

$$V_{вых} = f(V_{вх}), \qquad (3.42)$$

где $V_{вых}$ и $V_{вх}$ — сигналы на выходе и входе узла соответственно. Если узел имеет более чем один вход и один выход, то в (3.42) скаляры $V_{вых}$ и $V_{вх}$ становятся векторами.

Однако известно, что представление модели в виде (3.42) возможно только, если узел является безынерционным, т.е. в полной модели узла не фигурируют производные. Следовательно, для получения (3.42) в общем случае требуется предварительная алгебраизация полной модели. Такую алгебраизацию выполняют с помощью интегральных преобразований, например, с помощью преобразования Лапласа, переходя из временной области в пространство комплексной переменной p. Тогда в моделях типа (3.42) имеют место не оригиналы, а изображения сигналов $V_{вых}(p)$ и $V_{вх}(p)$, сами же модели реальных блоков стараются по возможности максимально упростить и представить их моделями типовых блоков (звеньев) из числа заранее разработанных библиотечных моделей. Обычно модели звеньев имеют вид

$$V_{вых}(p) = h(p)V_{вх}(p),$$

где $h(p)$ — передаточная функция звена.

В случае применения преобразования Лапласа появляются ограничения на использование нелинейных моделей, а именно: в моделях не должно быть нелинейных инерционных элементов.

Другое упрощающее допущение при моделировании на функционально-логическом уровне — неучет влияния нагрузки на характеристики блоков. Действительно, подключение к выходу блока не-

которого другого узла никак не влияет на модель блока (3.42).

Собственно получение ММС из ММЭ оказывается вследствие принятых допущений значительно проще, чем на макроуровне: ММС есть совокупность ММЭ, в которых отождествлены сигналы на соединенных входах и выходах элементов. Эта ММС представляет собой систему алгебраических уравнений.

Получение ММС проиллюстрируем простым примером (рис. 3.12), где показана система из трех блоков с передаточными функциями $h_1(p)$, $h_2(p)$ и $h_3(p)$. ММС имеет вид:

$$V_2 = h_1(p)V_1;$$
$$V_{вых}(p) = h_2(p)V_2;$$
$$V_1 = V_{вх}(p) + h_3(p)V_{вых}(p)$$

или

$$V_{вых}(p) = H(p)V_{вх}(p),$$

где $H(p) = h_1(p)\ h_2(p)\ /\ (1 - h_1(p)\ h_2(p)\ h_3(p))$

Рис. 3.12. Пример схемы из трех блоков

Таким образом, анализ сводится к следующим операциям:

1) заданную схему устройства представляют совокупностью звеньев и, если схема не полностью покрывается типовыми звеньями, то разрабатывают оригинальные модели;

2) формируют ММС из моделей звеньев;

3) применяют прямое преобразование Лапласа к входным сигналам;

4) решают систему уравнений ММС и находят изображения выходных сигналов;

5) с помощью обратного преобразования Лапласа возвращаются во временную область из области комплексной переменной p.

Математические модели дискретных устройств. Анализ дискретных устройств на функционально-логическом уровне требуется прежде всего при проектировании устройств вычислительной техники и цифровой автоматики. Здесь дополнительно к допущениям, принимаемым при анализе аналоговых устройств, используют дискретизацию сигналов, причем базовым является двузначное представление сигналов. Удобно этими двумя возможными значениями сигналов считать “истину”(иначе 1) и “ложь”(иначе 0), а сами сигналы рассматривать как булевы величины. Тогда для моделирования можно использовать аппарат математической логики. Находят применение также трех- и более значные модели. Смысл значений сигналов в многозначном моделировании и причины его применения будут пояснены ниже на некоторых примерах.

Элементами цифровых устройств на функционально-логическом уровне служат элементы, выполняющие логические функции и возможно функции хранения информации. Простейшими элементами являются дизъюнктор, конъюнктор, инвертор, реализующие соответственно операции дизъюнкции (ИЛИ) $y = a$ **or** b, конъюнкции (И) $y = a$ **and** b, отрицания (НЕ) $y =$ **not** a, где y- выходной сигнал, a и b — входные сигналы. Число входов может быть и более двух. Условные схемные обозначения простых логических элементов показаны на рис. 3.13.

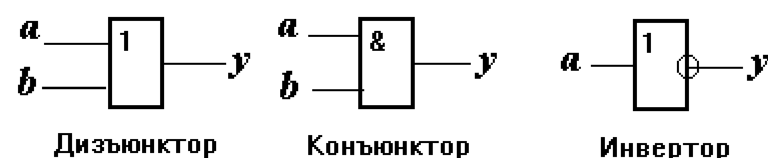


Рис. 3.13. Условные обозначения логических элементов на схемах

Математические модели устройств представляют собой систему математических моделей элементов, входящих в устройство, при отождествлении сигналов, относящихся к одному и тому же соединению элементов.

Различают синхронные и асинхронные модели.

Синхронная модель представляет собой систему логических уравнений, в ней отсутствует такая переменная как время, синхронные модели используют для анализа установившихся состояний.

Примером синхронной модели может служить следу-

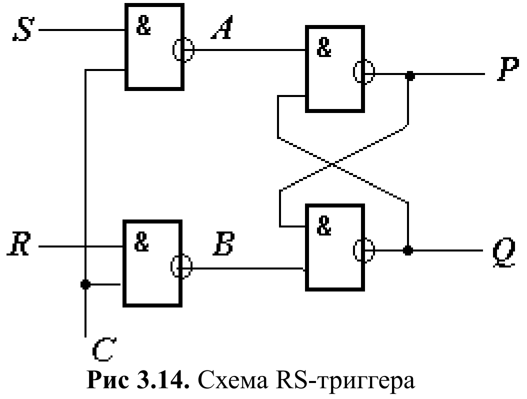


Рис 3.14. Схема RS-триггера

ющая система уравнений, полученная для логической схемы триггера (рис. 3.14):

B = **not** (R **and** C); Q = **not** (B **and** P); P = **not** (A **and** Q);

A = **not** (S **and** C).

Асинхронные модели отражают не только логические функции, но и временные задержки в распространении сигналов. Асинхронная модель логического элемента имеет вид

$$y(t+t_{зд}) = f(\mathbf{X}(t)), \qquad (3.43)$$

где $t_{зд}$ — задержка сигнала в элементе; f — логическая функция. Запись (3.43) означает, что выходной сигнал y принимает значение логической функции, соответствующее значениям аргументов $\mathbf{X}(t)$, в момент времени $t+t_{зд}$. Следовательно, асинхронные модели можно использовать для анализа динамических процессов в логических схемах.

Термины синхронная и асинхронная модели можно объяснить ориентированностью этих моделей на синхронные и асинхронные схемы соответственно. В синхронных схемах передача сигналов между цифровыми блоками происходит только при подаче на специальные синхровходы тактовых (синхронизирующих) импульсов. Частота тактовых импульсов выбирается такой, чтобы к моменту прихода синхроимпульса переходные процессы от предыдущих передач сигналов фактически закончились. Следовательно, в синхронных схемах расчет задержек не актуален, быстродействие устройства определяется заданием тактовой частоты.

Синхронные модели можно использовать не только для выявления принципиальных ошибок в схемной реализации заданных функций. С их помощью можно обнаруживать места в схемах, опасные, с точки зрения, возникновения в них искажающих помех. Ситуации, связанные с потенциальной опасностью возникновения помех и сбоев, называют *рисками сбоя*.

Различают статический и динамический риски сбоя. Статический риск сбоя иллюстрирует ситуация рис. 3.15, если на два входа элемента И могут приходить перепады сигналов в противоположных направлениях, как это показано на рис. 3.15,*б*. Если вместо идеального случая, когда оба перепада приходят в момент времени Т, перепады вследствие разброса задержек придут неодновременно, причем так, как показано на рис. 3.15,б, то на выходе элемента появляется импульс помехи, который может исказить работу всего устройства. Для устранения таких рисков сбоя нужно уметь их выявлять. С этой целью применяют трехзначное синхронное моделирование.

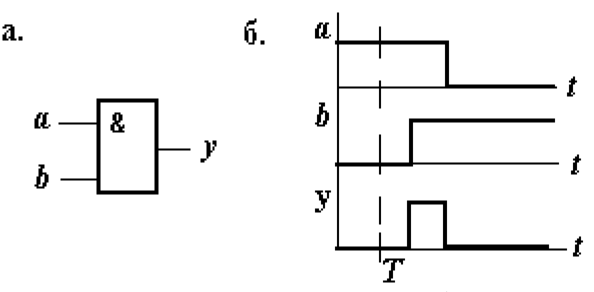


Рис. 3.15. Статический риск сбоя: **а** - схема; **б** - диаграмма сигналов

При этом тремя возможными значениями сигналов являются 0, 1 и ⊗, причем значение ⊗ интерпретируется как неопределенность. Правила выполнения логических операций И, ИЛИ, НЕ в трехзначном алфавите очевидны из рассмотрения табл. 3.6. В ней вторая строка отведена для значений одного аргумента, а первый столбец — для значений второго аргумента, значения функций представлены ниже второй строки и правее первого столбца.

Таблица 3.6.

	Операция		
	И	ИЛИ	НЕ
	0 ⊗ 1	0 ⊗ 1	0 ⊗ 1
0	0 0 0	0 ⊗ 1	1 ⊗ 0
⊗	0 ⊗ ⊗	⊗ ⊗ 1	-
1	0 ⊗ 1	1 1 1	-

При анализе рисков сбоя на каждом такте вместо однократного решения уравнений модели производят двукратное решение, поэтому можно говорить об исходных, промежуточных (после первого решения) и итоговых (после второго решения) значениях переменных. Для входных сигналов допустимы только такие последовательности исходных, промежуточных и итоговых значений: 0-0-0, 1-1-1, 0-⊗-1, 1-⊗-0. Для других переменных появление последовательности 0-⊗-0 или 1-⊗-1 означает неопределенность во время переходного процесса, т.е. возможность статического риска сбоя.

Для простейшей схемы (рис. 3.15,*а*) результаты трехзначного моделирования представлены в табл. 3.7.

Таблица 3.7

Значения	*a*	*B*	*y*
исходные	1	0	0
промежуточные	⊗	⊗	⊗
итоговые	0	1	0

Динамический риск сбоя иллюстрируют схема и временные диаграммы рис. 3.16. Сбой выражается в появлении вместо одного перепада на выходе, что имеет место при правильном функционировании, нескольких перепадов. Обнаружение динамических рисков сбоя также выполняют с помощью двукратного решения уравнений модели, но при использовании пятизначного алфавита с множеством значений {0, 1, ⊗, α, β}, где α интерпретируется как положительный перепад, β — как отрицательный перепад, остальные символы имеют прежний смысл. В отсутствие сбоев последовательности значений переменных в исходном, промежуточном и итоговом состояниях могут быть такими: 0-0-0, 1-1-1, 0-α-1, 1-β-0. Последовательности 0-⊗-1 или 1-⊗-0 указывают на динамический риск сбоя.

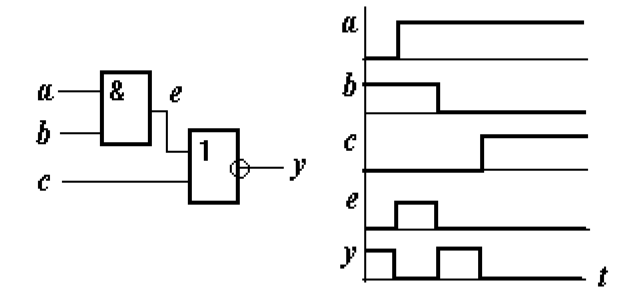


Рис. 3.16. Динамический риск сбоя

Трехзначный алфавит можно использовать и в асинхронных моделях. Пусть в модели $y(t+t_{зд}) = f(\mathbf{X}(t))$ в момент времени t_1 входы $\mathbf{X}(t_1)$ таковы, что в момент времени $t_1+t_{зд}$ происходит переключение выходного сигнала y. Но если учитывать разброс задержек, то $t_{зд}$ принимает некоторое случайное значение в диапазоне $[t_{зд\ min}, t_{зд\ max}]$ и, следовательно, в модели в интервале времени от $t_1+t_{зд\ min}$ до $t_1+t_{зд\ max}$ сигнал y должен иметь неопределенное значение Д. Именно это и достигается с помощью трехзначного асинхронного моделирования.

Методы логического моделирования. В отношении асинхронных моделей возможны два метода моделирования — пошаговый (инкрементный) и событийный.

В *пошаговом методе* время дискретизируется и вычисления по выражениям модели выполняются в дискретные моменты времени t_0, t_1, t_2... и т.д. Шаг дискретизации ограничен сверху значением допустимой погрешности определения задержек и потому оказывается довольно малым, а время анализа значительным.

Для сокращения времени анализа используют *событийный метод*. В этом методе событием называют изменение любой переменной модели. Событийное моделирование основано на следующем правиле: *обращение к модели логического элемента происходит только в том случае, если на входах этого элемента произошло событие*. В сложных логических схемах на каждом такте синхронизации обычно происходит переключение всего лишь 2-3% логических элементов и, соответственно, в событийном методе в несколько раз уменьшаются вычислительные затраты по сравнению с пошаговым моделированием.

Методы анализа синхронных моделей представляют собой методы решения систем логических уравнений. К этим методам относятся метод простых итераций и метод Зейделя, которые аналогичны одноименным методам решения систем алгебраических уравнений в непрерывной математике.

Применение этих методов к моделированию логических схем удобно проиллюстрировать на примере схемы триггера (см. рис. 3.14). В табл. 3.8 представлены значения переменных модели в исходном состоянии и после каждой итерации в соответствии с методом простых итераций. В исходном состоянии задают начальные (возможно произвольные) значения промежуточных и выходных переменных, в данном примере это значения переменных *B, Q, P, A*, соответствующие предыдущему состоянию триггера . Новое состояние триггера должно соответствовать указанным в таблице изменив-

Таблица 3.8

Итерация	*R*	*S*	*C*	*B*	*Q*	*P*	*A*
Предыдущее состояние	0	0	0	1	1	0	1
Исходные значения (итерация 0)	0	1	1	1	1	0	1
Итерация 1	0	1	1	1*	1	0	0*
Итерация 2	0	1	1	1	1	1*	0
Итерация 3	0	1	1	1	0*	1	0
Итерация 4	0	1	1	1	0	1*	0

шимся значениям входных сигналов *R, S* и *C*. Вычисления заканчиваются, если на очередной итерации изменений переменных нет, что и наблюдается в данном примере на четвертой итерации.

Согласно методу простых итераций, в правые части уравнений модели на каждой итерации подставляют значения переменных, полученные на предыдущей итерации. В отличие от этого в методе Зейделя, если у некоторой переменной обновлено значение на текущей итерации, то именно его и используют в дальнейших вычислениях уже на текущей итерации. Метод Зейделя позволяет сократить число итераций, но для этого нужно предварительно упорядочить уравнения модели так, чтобы последовательность вычислений соответствовала последовательности прохождения сигналов по схеме. Такое упорядочение выполняют с помощью ранжирования.

Ранжирование заключается в присвоении элементам и переменным модели значений рангов в соответствии со следующими правилами: 1) в схеме разрываются все контуры обратной связи, что приводит к появлению дополнительных входов схемы (псевдовходов); 2) все внешние переменные (в том числе на псевдовходах) получают ранг 0; 3) элемент и его выходные переменные получают ранг *k*, если у элемента все входы проранжированы и старший среди рангов входов равен *k*-1.

Так, если в схеме (см. рис. 3.14) разорвать имеющийся контур обратной связи в цепи переменной *Q* и обозначить переменную на псевдовходе Q_1, то ранги переменных оказываются следующими: *R, S, C,* Q_1 имеют ранг 0, *A* и *B* — ранг 1, *P* — ранг 2 и *Q* — ранг 3. В соответствии с этим переупорядочивают уравнения в модели триггера:

A = **not** (*S* **and** *C*). *B* = **not** (*R* **and** *C*); *P* = **not** (*A* **and** *Q*); *Q* = **not** (*B* **and** *P*).

Теперь уже на первой итерации по Зейделю получаем требуемый результат. Если разорвать контур обратной связи в цепи переменной P, то решение в данном примере будет получено после второй итерации, но это все равно заметно быстрее, чем при использовании метода простой итерации.

Для сокращения объема вычислений в синхронном моделировании возможно использование событийного подхода. По-прежнему обращение к модели элемента происходит, только если на его входах произошло событие.

Для триггера (см. рис. 3.14) применение событийности в рамках метода простых итераций приводит к сокращению объема вычислений: вместо 16-кратных обращений к моделям элементов, как это видно из табл. 3.8, происходит лишь 5-кратное обращение. В табл. 3.8 звездочками помечены значения переменных, вычисляемые в событийном методе. Так, например, на итерации 0 имеют место изменения переменных *S* и *C*, поэтому на следующей итерации обращения происходят только к моделям элементов с выходами *A* и *B*.

3.6. Математическое обеспечение анализа на системном уровне

Основные сведения из теории массового обслуживания. Объектами проектирования на системном уровне являются такие сложные системы, как производственные предприятия, транспортные системы, вычислительные системы и сети, автоматизированные системы проектирования и управления и т.п. В этих приложениях анализ процессов функционирования систем связан с исследованием прохождения через систему потока *заявок* (иначе называемых *требованиями* или *транзактами*). Разработчиков подобных сложных систем интересуют прежде всего такие параметры, как производительность (пропускная способность) проектируемой системы, продолжительность обслуживания (задержки) заявок в системе, эффективность используемого в системе оборудования.

Заявками могут быть заказы на производство изделий, задачи, решаемые в вычислительной системе, клиенты в банках, грузы, поступающие на транспортировку и др. Очевидно, что параметры заявок, поступающих в систему, являются случайными величинами и при проектировании могут быть известны лишь их законы распределения и числовые характеристики этих распределений. Поэтому анализ функционирования на системном уровне, как правило, носит статистический характер. В качестве математического аппарата моделирования удобно принять теорию массового обслуживания, а в качестве моделей систем на этом уровне использовать *системы массового обслуживания* (СМО).

Типичными выходными параметрами в СМО являются числовые характеристики таких величин, как время обслуживания заявок в системе, длины очередей заявок на входах, время ожидания обслуживания в очередях, загрузка устройств системы, а также вероятность обслуживания в заданные сроки и т.п.

В простейшем случае СМО представляет собой некоторое средство (устройство), называемое *обслуживающим аппаратом* (ОА), вместе с очередями заявок на входах. Более сложные СМО состо-

ят из многих взаимосвязанных ОА. Обслуживающие аппараты СМО в совокупности образуют *статические объекты* СМО, иначе называемые *ресурсами*. Например, в вычислительных сетях ресурсы представлены аппаратными и программными средствами.

В СМО, кроме статических объектов, фигурируют *динамические объекты* — транзакты. Например, в вычислительных сетях динамическими объектами являются решаемые задачи и запросы на информационные услуги.

Состояние СМО характеризуется состояниями составляющих ее объектов. Например, состояния ОА выражаются булевыми величинами, значения которых интерпретируются как true (занято) и false (свободно), и длинами очередей на входах ОА, принимающими неотрицательные целочисленные значения. Переменные, характеризующие состояние СМО, будем называть переменными состояния или фазовыми переменными.

Правило, согласно которому заявки выбирают из очередей на обслуживание, называют *дисциплиной обслуживания*, а величину, выражающую преимущественное право на обслуживание, — *приоритетом*. В *бесприоритетных дисциплинах* все транзакты имеют одинаковые приоритеты. Среди бесприоритетных дисциплин наиболее популярны дисциплины FIFO (первым пришел — первым обслужен), LIFO (последним пришел — первым обслужен) и со случайным выбором заявок из очередей.

В *приоритетных дисциплинах* для заявок каждого приоритета на входе ОА выделяется своя очередь. Заявка из очереди с низким приоритетом поступает на обслуживание, если пусты очереди с более высокими приоритетами. Различают приоритеты абсолютные, относительные и динамические. Заявка из очереди с более высоким *абсолютным приоритетом*, поступая на вход занятого ОА, прерывает уже начатое обслуживание заявки более низкого приоритета. В случае *относительного приоритета* прерывания не происходит, более высокоприоритетная заявка ждет окончания уже начатого обслуживания. Динамические приоритеты могут изменяться во время нахождения заявки в СМО.

Исследование поведения СМО, т.е. определение временных зависимостей переменных, характеризующих состояние СМО, при подаче на входы любых требуемых в соответствии с заданием на эксперимент потоков заявок, называют *имитационным моделированием* СМО. Имитационное моделирование проводят путем воспроизведения событий, происходящих одновременно или последовательно в модельном времени. При этом под *событием* понимают факт изменения значения любой фазовой переменной.

Подход, альтернативный имитационному моделированию, называют *аналитическим исследованием* СМО. Аналитическое исследование заключается в получении формул для расчета выходных параметров СМО с последующей подстановкой значений аргументов в эти формулы в каждом отдельном эксперименте.

Модели СМО, используемые при имитационном и аналитическом моделировании, называются имитационными и аналитическими соответственно.

Аналитические модели удобны в использовании, поскольку для аналитического моделирования не требуются сколько-нибудь значительные затраты вычислительных ресурсов, часто без постановки специальных вычислительных экспериментов разработчик может оценить характер влияния аргументов на выходные параметры, выявить те или иные общие закономерности в поведении системы. Но, к сожалению, аналитическое исследование удается реализовать только для частных случаев сравнительно несложных СМО. Для сложных СМО аналитические модели если и удается получить, то только при принятии упрощающих допущений, ставящих под сомнение адекватность модели.

Поэтому основным подходом к анализу САПР на системном уровне проектирования считают имитационное моделирование, а аналитическое исследование используют при предварительной оценке различных предлагаемых вариантов систем.

Некоторые компоненты СМО характеризуются более чем одним входным и (или) выходным потоками заявок. Правила выбора одного из возможных направлений движения заявок входят в соответствующие модели компонентов. В одних случаях такие правила относятся к исходным данным (например, выбор направления по вероятности), но в некоторых случаях желательно найти оптимальное управление потоками в узлах разветвления. Тогда задача моделирования становится более сложной задачей синтеза, характерными примерами являются маршрутизация заявок или синтез расписаний и планов.

Аналитические модели СМО. Как отмечено выше, аналитические модели СМО удается получить при довольно серьезных допущениях. К числу типичных допущений относятся следующие.

Во-первых, как правило, считают, что в СМО используются бесприоритетные дисциплины обслуживания типа FIFO.

Во-вторых, времена обслуживания заявок в устройствах выбираются в соответствии с экспоненциальным законом распределения.

В-третьих, в аналитических моделях СМО входные потоки заявок аппроксимируются *простейшими* потоками, т.е. потоками, обладающими свойствами стационарности, ординарности (невозможности одновременного поступления двух заявок на вход СМО), отсутствия последействия.

В большинстве случаев модели СМО отображают процессы с конечным множеством состояний и с отсутствием последействия. Такие процессы называют *конечными марковскими цепями*.

Марковские цепи характеризуются множеством состояний **S**, матрицей вероятностей переходов из одного состояния в другое и начальными условиями (начальным состоянием). Удобно представлять марковскую цепь в виде графа, в котором вершины соответствуют состояниям цепи, дуги — переходам, веса дуг — вероятностям переходов (если время дискретно) или интенсивностям переходов (если время непрерывно).

Отметим, что интенсивностью перехода называют величину $V_{ij} = \lim P_{ij}(t_1) / t_1$ при $t_1 \to 0$, где $P_{ij}(t_1)$ — вероятность перехода из состояния S_i в состояние S_j за время t_1. Обычно принимается условие

$$V_{ii} = -\sum_{j \neq i} V_{ij},$$

что означает

$$\sum_{j=1}^{N} V_{ij} = 0. \qquad (3.44)$$

где N — число состояний. На рис. 3.17 приведен пример марковской цепи в виде графа с состояниями $S_1,...,S_4$, а в табл. 3.9 представлена матрица интенсивностей переходов для этого примера.

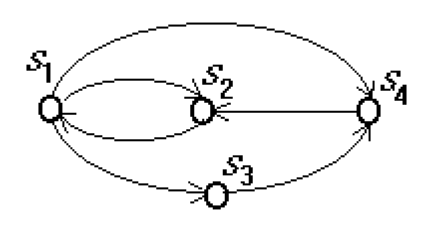


Рис.3.17. Пример марковской цепи

Таблица 3.9

Состояние	S_1	S_2	S_3	S_4
S_1	$-V_{12}-V_{13}-V_{14}$	V_{12}	V_{13}	V_{14}
S_2	V_{21}	$-V_{21}$	0	0
S_3	0	0	$-V_{34}$	V_{34}
S_4	0	V_{42}	0	$-V_{42}$

Большинство выходных параметров СМО можно определить, используя информацию о поведении СМО, т.е. информацию о состояниях СМО в установившихся (стационарных) режимах и об их изменениях в переходных процессах. Эта информация имеет вероятностную природу, что обусловливает описание поведения СМО в терминах вероятностей нахождения системы в различных состояниях. Основой такого описания, а следовательно, и многих аналитических моделей СМО являются *уравнения Колмогорова*.

Уравнения Колмогорова можно получить следующим образом.

Изменение вероятности P_i нахождения системы в состоянии S_i за время t_1 есть вероятность перехода системы в состояние S_i из любых других состояний за вычетом вероятности перехода из состояния S_i в другие состояния за время t_1, т.е.

$$P_i(t) = P_i(t+t_1) - P_i(t) = \sum_{j \in \mathbf{J}} P_{ji}(t_1)P_j(t) - \sum_{k \in \mathbf{K}} P_{ik}(t_1)P_i(t), \qquad (3.45)$$

где $P_i(t)$ и $P_j(t)$ — вероятности нахождения системы в состояниях S_i и S_j соответственно в момент времени t, а $P_{ji}(t_1)$ и $P_{ik}(t_1)$ — вероятности изменения состояний в течение времени t_1; произведение вида

$P_{ji}(t_1)P_j(t)$ есть безусловная вероятность перехода из S_j в S_i, равная условной вероятности перехода, умноженной на вероятность условия; **J** и **K** — множества индексов инцидентных вершин по отношению к вершине S_i по входящим и исходящим дугам на графе состояний соответственно.

Разделив выражение (3.45) на t_1 и перейдя к пределу при $t \to 0$, получим

$$\lim_{t\to 0} P_i(t)/t_1 = \sum_j (\lim_{t\to 0} P_{ji}/t_1)P_j - \sum_k (\lim_{t\to 0} P_{ik}/t_1)P_i,$$

откуда следуют уравнения Колмогорова

$$dP_i/dt = \sum_j (V_{ji} P_j) - P_i \sum_k V_{ik}.$$

В стационарном состоянии $dP_i/dt = 0$ и уравнения Колмогорова составляют систему алгебраических уравнений, в которой i-й узел представлен уравнением

$$\sum_j (V_{ji}P_j) = P_i \sum_k V_{ik}. \quad (3.46)$$

Прибавляя $V_{ii}P_i$ к левой и правой частям уравнения (3.46) и учитывая (3.44), получаем

$$\sum_{j=1}^{N} (V_{ji}P_j) = P_i \sum_{k=1}^{N} V_{ik} = 0,$$

т.е.

$$\sum_{j=1}^{N} (V_{ji}P_j) = 0,$$

где P_j — финальные вероятности.

Пример аналитической модели. Примером СМО, к которой можно применить аналитические методы исследования, является одноканальная СМО с простейшим входным потоком интенсивностью λ и длительностью обслуживания, подчиняющейся экспоненциальному закону обслуживания интенсивностью μ. Для этой СМО нужно получить аналитические зависимости среднего числа N_{av} заявок, находящихся в системе, среднюю длину Q_{av} очереди к ОА, время T_{av} пребывания заявки в системе, время T_{or} ожидания в очереди.

На рис. 3.18 представлен граф состояний рассматриваемой СМО, где S_k — состояние с k заявками в системе. Матрица интенсивностей представлена в табл. 3.10. Уравнения Колмогорова для установившегося режима имеют вид

$$\lambda P_0 + \mu P_1 = 0,$$
$$\lambda P_0 - (l+\mu)P_1 + \mu P_2 = 0,$$
$$\mu P_1 - (l+\mu)P_2 + \lambda P_3 = 0,$$
$$\mu P_2 - (l+\mu)P_3 + \lambda P_4 = 0,$$

.....

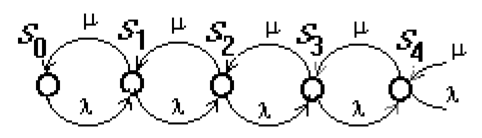


Рис.3.18. Граф состояний

Таблица 3.10

Состояние	S_0	S_1	S_2	S_3	S_4	...
S_0	-λ	λ	0	0	0	...
S_1	μ	-λ-μ	λ	0	0	...
S_2	0	μ	-λ-μ	λ	0	...
S_3	0	0	μ	-λ-μ	λ	...
S_4	0	0	0	μ	-λ-μ	...
...	...	...	...	...	...	...

Используя уравнения Колмогорова, можно выразить все P_l, $i = 1,2,3...$, через P_0. Получим

$$P_1 = \lambda P_0/\mu = aP_0,$$
$$P_2 = ((l+\mu)P_1 - \lambda P_0) / \mu = (1+a)P_1 - aP_0 = a^2P_0;$$
$$P_3 = (1+a)P_2 - aP_1 = a^2P_1 = a^3P_0 \text{ и т.д.}$$

Здесь введено обозначение $a = \lambda/\mu$. Отметим также, что установившийся режим возможен только при $a < 1$.

Так как $\sum_{i=0}^{\infty} P_i = 1$, то

$$P_0 = 1 - \sum_{i=0}^{\infty} P_i = 1 - P_0(a + a^2 + a^3 + ...) = 1 / (1 + a + a^2 + a^3 + ...) = 1 - a.$$

Теперь нетрудно получить и остальные требуемые результаты:

$$N_{av} = \sum_{k=1}^{\infty} P_k k = P_1 + 2P_2 + 3P_3 + ... = a\,(1-a)\,(1 + 2a + 3a^2 + ...) = a\,(1-a) / (1-a)^2 = a / (1-a).$$

$$Q_{av} = P_2 + 2P_3 + 3P_4 + ... = \sum_{k=2}^{\infty} (k-1)P_k = P_0\, a^2 (1 + 2a + 3a^2 + ...) = a^2 / (1-a).$$

Времена пребывания в системе и очереди находятся из соотношений:

$$N_{av} = \lambda T_{av},$$
$$Q_{av} = \lambda T_{or}$$

которые называют формулами Литтла:

$$T_{av} = a / (1-a) / \lambda = 1 / (\mu - \lambda),$$
$$T_{or} = a^2 / (1-a) / \lambda = a / (\mu - \lambda).$$

Имитационное моделирование СМО. Для представления имитационных моделей можно использовать языки программирования общего применения, однако такие представления оказываются довольно громоздкими. Поэтому обычно применяют специальные языки имитационного моделирования на системном уровне. Среди языков имитационного моделирования различают языки, ориентированные на описание событий, средств обслуживания или маршрутов движения заявок (процессов). Выбор языка моделирования определяет структуру модели и методику ее построения.

Ориентация на устройства характерна для функционально-логического и более детальных иерархических уровней описания объектов.

Для описания имитационных моделей на системном уровне (такие модели иногда называют *сетевыми имитационными моделями* — СИМ) чаще используют языки, ориентированные на события или процессы. Примерами первых могут служить языки Симскрипт, SMPL и ряд других. К числу вторых относятся языки Симула, SOL, а также популярный язык GPSS.

Языки имитационного моделирования реализуются в программно-методических комплексах моделирования СМО, имеющих ту или иную степень специализации. Так, комплексы на базе языка GPSS можно использовать во многих приложениях, но есть специализированные комплексы для моделирования вычислительных сетей, систем управления предприятиями и т.п.

При использовании языков, ориентированных на процессы, в составе СИМ выделяются элементарные части и ими могут быть источники входных потоков заявок, устройства, накопители и узлы. *Источник входного потока заявок* представляет собой алгоритм, в соответствии с которым вычисляются моменты t_k появления заявок на выходе источника. Источники могут быть зависимыми и независимыми. В зависимых источниках моменты появления заявок связаны с наступлением определенных событий, например, с приходом другой заявки на вход некоторого устройства. Типичным независимым источником является алгоритм выработки значений t_k случайной величины с заданным законом распределения.

Устройства в имитационной модели представлены алгоритмами выработки значений интервалов (длительностей) обслуживания. Чаще всего это алгоритмы генерации значений случайных величин с заданным законом распределения. Но могут быть устройства с детерминированным временем обслуживания или временем, определяемым событиями в других частях СИМ. Модель устройства отображает также заданную дисциплину обслуживания, поскольку в модель входит алгоритм, управляющий очередями на входах устройства.

Накопители моделируются алгоритмами определения объемов памяти, занимаемых заявками, приходящими на вход накопителя. Обычно объем памяти, занимаемый заявкой, вычисляется как зна-

чение случайной величины, закон и (или) числовые характеристики распределения может зависеть от типа заявки.

Узлы выполняют связующие, управляющие и вспомогательные функции в имитационной модели, например, для выбора направлений движения заявок в СИМ, изменения их параметров и приоритета, разделения заявок на части, их объединения и т.п.

Обычно каждому типу элементарной модели, за исключением лишь некоторых узлов, в программной системе соответствует определенная процедура (подпрограмма). Тогда СИМ можно представить как алгоритм, состоящий из упорядоченных обращений к этим процедурам, отражающим поведение моделируемой системы.

В процессе моделирования происходят изменения модельного времени, которое чаще всего принимается дискретным, измеряемым в тактах. Время изменяется после того, как закончена имитация очередной группы событий, относящихся к текущему моменту времени t_k. Имитация сопровождается накоплением в отдельном файле статистики таких данных, как количества заявок, вышедших из системы обслуженными и необслуженными, суммарное время занятого состояния для каждого из устройств, средние длины очередей и т.п. Имитация заканчивается, когда текущее время превысит заданный отрезок времени или когда входные источники выработают заданное число заявок. После этого производят обработку накопленных в файле статистики данных, что позволяет получить значения требуемых выходных параметров.

Событийный метод моделирования. В программах имитационного моделирования СМО преимущественно реализуется *событийный метод* организации вычислений. Сущность событийного метода заключается в отслеживании на модели последовательности событий в том же порядке, в каком они происходили бы в реальной системе. Вычисления выполняют только для тех моментов времени и тех частей (процедур) модели, к которым относятся совершаемые события. Другими словами, обращения на очередном такте моделируемого времени осуществляются только к моделям тех элементов (устройств, накопителей), на входах которых в этом такте произошли изменения. Поскольку изменения состояний в каждом такте обычно наблюдаются лишь у малой доли ОА, событийный метод может существенно ускорить моделирование по сравнению с инкрементным методом, в котором на каждом такте анализируются состояния всех элементов модели.

Рассмотрим возможную схему реализации событийного метода имитационного моделирования.

Моделирование начинается с просмотра операторов генерирования заявок, т.е. с обращения к моделям источников входных потоков. Для каждого независимого источника такое обращение позволяет рассчитать момент генерации первой заявки. Этот момент вместе с именем — ссылкой на заявку — заносится в список будущих событий (СБС), а сведения о генерируемой заявке — в список заявок (СЗ). Запись в СЗ включает в себя имя заявки, значения ее параметров (атрибутов), место, занимаемое в данный момент в СИМ. В СБС события упорядочиваются по увеличению моментов наступления.

Затем из СБС выбирают совокупность сведений о событиях, относящихся к наиболее раннему моменту времени. Эта совокупность переносится в список текущих событий (СТС), из которого извлекаются ссылки на события. Обращение по ссылке к СЗ позволяет установить место в СИМ заявки А, с которой связано моделируемое событие. Пусть этим местом является устройство Х. Далее программа моделирования выполняет следующие действия (рис. 3.19):

1) изменяет параметры состояния устройства Х; например, если заявка А освобождает Х, а очередь к Х не была пуста, то в соответствии с заданной дисциплиной обслуживания из очереди к Х выбирается заявка В и поступает на обслуживание в Х;

2) прогнозируется время наступления следующего события, связанного с заявкой В, путем обращения к модели устройства Х,

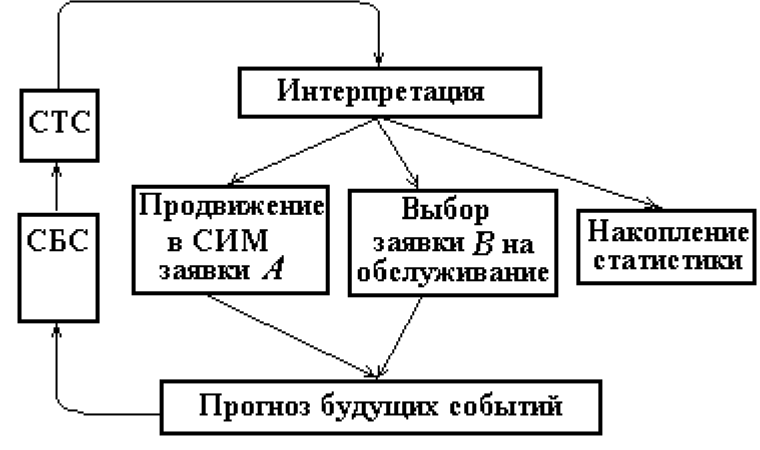


Рис. 3.19. Иллюстрация событийного моделирования

в которой рассчитывается продолжительность обслуживания заявки В; сведения об этом будущем событии заносятся в СБС и СЗ;

3) происходит имитация движения заявки А в СИМ по маршруту, определяемому заданной программой моделирования, до тех пор, пока заявка не придет на вход некоторого ОА; здесь либо заявка задерживается в очереди, либо путем обращения к модели этого ОА прогнозируется наступление некоторого будущего события, связанного с дальнейшей судьбой заявки А; сведения об этом будущем событии также заносятся в СБС и СЗ;

4) в файл статистики добавляются необходимые данные.

После отработки всех событий, относящихся к моменту времени t_k, происходит увеличение модельного времени до значения, соответствующего ближайшему будущему событию, и рассмотренный процесс имитации повторяется.

Краткое описание языка GPSS. Язык GPSS (General Purpose Simulation System), ориентированный на процессы, реализован в ряде программ имитационного моделирования. Модель (программа) на языке GPSS представляет собой последовательность операторов (их называют блоками), отображающих события, происходящие в СМО при перемещениях транзактов. Поскольку в интерпретаторах GPSS реализуется событийный метод, и в СМО может быть одновременно много транзактов, то интерпретатор будет попеременно исполнять разные фрагменты программы, имитируя продвижения транзактов в текущий момент времени до их задержки в некоторых устройствах или очередях.

Операторы GPSS имеют следующий формат:

```
<метка> <имя оператора> <поле операндов> [<комментарий>]
```

причем метка может занимать позиции, начиная со второй, имя оператора — с восьмой, поле операндов — с девятнадцатой, комментарий обязательно отделяется от поля операндов пробелом.

Поле операндов может быть пусто, иметь один или более операндов, обозначаемых ниже при описании блоков символами A, B, C,... Операндами могут быть идентификаторы устройств, накопителей, служебные слова и стандартные числовые атрибуты (СЧА). К СЧА относятся величины, часто встречающиеся в разных задачах. Это, например, AC1 — текущее время, FN — функция, P — параметр транзакта (каждый транзакт может иметь не более L параметров, обычно L =12), K — константа, RN1 — случайная величина, равномерно распределенная в диапазоне [0, 1], S — объем занятой памяти в накопителе, F — состояние устройства, Q — текущая длина очереди и др. При этом ссылки на идентификаторы записываются в виде

```
<СЧА>$<идентификатор>
```

например, Q$ORD означает очередь ORD или FN$COS — ссылка на функцию COS.

Рассмотрим наиболее часто встречающиеся операторы, сопровождая знакомство с ними простыми примерами моделей.

Источники заявок обычно описываются блоком

```
GENERATE A,B,C,D,E
```

где A и B служат для задания интервалов между появлениями заявок, при этом можно использовать один из следующих вариантов: 1) интервал — равномерно распределенная в диапазоне [A-B, A+B] случайная величина; 2) интервал — значение функции, указанной в B, умноженной на A; C — задержка в выработке первого транзакта; D — число вырабатываемых источником заявок; E — приоритет заявок. Если D пусто, то число вырабатываемых транзактов неограничено. Например:

```
GENERATE 6,FN$EXP,,15
```

Этот оператор описывает источник, который вырабатывает 15 транзактов с интервалами, равными произведению числа 6 и значения функции EXP;

```
GENERATE 36,12
```

Здесь число транзактов неограничено, интервалы между транзактами — случайные числа в диапазоне [24, 48].

Функции, на которые имеются ссылки в операторах должны быть описаны с помощью блока следующего типа

```
M FUNCTION A,B
```

за которым следует строка, начинающаяся с первой позиции

```
X1,Y1/X2,Y2/X3,Y3/..../Xn,Yn
```

Здесь метка M — идентификатор функции, A — аргумент функции, B — тип функции, X_i и Y_i — координаты узловых точек функции, заданной таблично. Например:

```
EXP FUNCTION RN1,C12
0,0/0.2,0.22/0.4,0.51/0.5,0.6/0.6,0.92/...
```
и т.д.

Это описание непрерывной (C) функции EXP, заданной таблично 12-ю узловыми точками, аргументом является случайная величина (RN1), равномерно распределенная в диапазоне [0, 1]; или

```
BBB FUNCTION *4,D6
1,2/2,5/3,11/4,20/5,18/6,12/7,9
```

Дискретная (D) функция BBB задана 6-ю узловыми точками, аргумент — четвертый параметр транзакта, возбудивший обращение к функции BBB.

Тразакты могут порождаться и оператором размножения

```
SPLIT A,B,C
```

когда в него входит некоторый транзакт. При этом создается семейство транзактов, включающее основной (вошедший в блок) транзакт и A его копий. Основной транзакт переходит в следующий по порядку блок, а его копии переходят в блок с меткой B. Для различения транзактов параметр C основного транзакта увеличивается на 1, а транзактов-копий — на 2, 3, 4,... и т.д.

Обратное действие — сборка транзактов выполняется операторами

```
ASSEMBLE A
```

согласно которому первый из вошедших в блок транзактов выйдет из него только после того, как в этот блок придут еще A-1 транзактов того же семейства, или оператором

```
GATHER A
```

отличающимся от предыдущего оператора тем, что из блока выходят все A транзактов.

Оператор

```
SEIZE A
```

описывает занятие устройства A транзактом, а оператор

```
RELEASE A
```

освобождение устройства A от обслуживания.

Задержка в движении транзакта по СМО описывается оператором

```
ADVANCE A,B
```

где A и B имеют тот же смысл, что и в операторе GENERATE.

Пример 1. Обслуживание транзакта в устройстве WST продолжительностью a единиц времени, где a — равномерно распределенная в диапазоне [7,11] случайная величина, описывается следующим фрагментом программы

```
...
SEIZE WST
ADVANCE 9,2
RELEASE WST
...
```

Аналогично описывается занятие транзактом памяти в накопителе

```
ENTER A,B
```

за исключением того, что здесь помимо имени накопителя (A) указывается объем занимаемой памяти (B). Освобождение B ячеек памяти в накопителе A выполняется оператором

```
LEAVE A,B
```

Для накопителей в модели нужно задавать общий объем памяти, что делается в следующем описании накопителя

```
M STORAGE A
```

где M — имя накопителя, A — объем памяти.

Если транзакт приходит на вход занятого устройства или на вход накопителя с недостаточным объемом свободной памяти, то он задерживается в очереди к этому устройству или накопителю. Слежение за состоянием устройств и очередей выполняет интерпретатор. Но если в модели требуется ссылаться на длину очереди или собирать статистику по ее длине, то нужно явное указание этой очереди в модели. Делается это с помощью операторов входа в очередь

```
QUEUE A
```

и выхода из очереди

```
DEPART A
```

согласно которым очередь A увеличивается и уменьшается на единицу соответственно.

Движение транзактов выполняется в естественном порядке, изменение этого порядка производится операторами перехода. Оператор условного перехода

```
TEST XX A,B,C
```

В соответствии с которым переход к оператору, помеченному меткой C, происходит, если не выполняется условие A XX B, где XX O {E,NE,L,LE,G,GE}, E- равно, NE — неравно, L — меньше, LE — меньше или равно, G — больше, GE — больше или равно (XX размещается в позициях 13 и 14).

Пример 2. Приходящие пользователи ожидают обслуживания, если длина очереди не более 4, иначе от обслуживания отказываются. Соответствующий фрагмент программы

```
...
TEST LE Q$STR,K4,LBL
QUEUE STR
SEIZE POINT
DEPART STR
ADVANCE 50,16
RELEASE POINT
...
LBL TERMINATE 1
...
```

В примере 2 использован оператор выхода транзактов из СМО

```
TERMINATE A
```

согласно которому из итогового счетчика вычитается число A.

С помощью итогового счетчика задается длительность моделирования. В начале исполнения программы в счетчик заносится число, указанное в операнде A оператора

```
START A,,C
```

Моделирование прекращается, когда содержимое счетчика будет равно или меньше нуля. Операнд C — шаг вывода статистики на печать.

Пример 3. Общая структура программы на GPSS имеет вид

```
SIMULATE
<описания, в том числе функций и накопителей >
<операторы, моделирующие движение транзактов>
```

```
START A,,C
END.
```

Оператор безусловного перехода

```
TRANSFER ,B
```

где B — метка оператора, к которому следует переход.

Используется ряд других разновидностей оператора TRANSFER. Например:

```
TRANSFER P,B,C
```

Переход происходит к оператору с меткой, равной сумме значения параметра B транзакта и числа C.

```
TRANSFER FN,B,C
```

То же, но вместо параметра транзакта слагаемым является значение функции B.

```
TRANSFER PICK,B,C
```

Это оператор равновероятного перехода к операторам, метки которых находятся в интервале [B,C]. Важное место в СМО занимает переход по вероятности

```
TRANSFER A,B,C
```

где A — вероятность перехода к оператору с меткой C, переход к оператору с меткой B будет происходить с вероятностью 1 - A.

Пример 4. Заказы, поступающие в СМО в случайные моменты времени в диапазоне [20,40], выполняет сначала бригада WGR1, затем параллельно работают бригады WGR2 и WGR3, каждая над своей частью заказа. Заданы экспоненциальные законы для времен выполнения работ бригадами WGR1, WGR2 и WGR3 с интенсивностями 0,05, 0,1 и 0,125 соответственно. Моделирование нужно выполнить на временном отрезке, соответствующем выполнению 1000 заказов.

Программа:

```
SIMULATE
EXP FUNCTION RN1,C12
0,0/.2,.22//.4,.51/.5,.6/.6,.92/.7,1.2/.8,1.61/.9,2.3/.95,3/.99,4.6/.999,6.9/1,1000
GENERATE 30,10
SEIZE WGR1
ADVANCE 20,FN$EXP
RELEASE WGR1
SPLIT 1,MET1
SEIZE WGR2
ADVANCE 10,FN$EXP
RELEASE WGR2
TRANSFER , MET2
MET1 SEIZE WGR3
ADVANCE 8,FN$EXP
RELEASE WGR3
MET2 ASSEMBLE 2
TERMINATE 1
START 1000,,1000
END
```

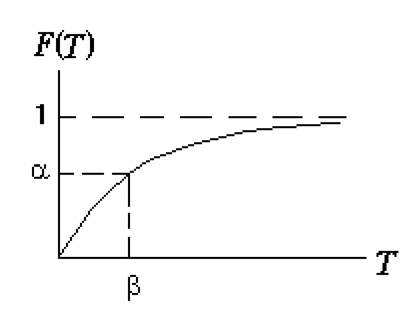


Рис. 3.20. Функция экспоненциального закона распределения

В этом примере использован экспоненциальный закон распределения с плотностью

$$P(t) = \lambda \exp(-\lambda T),$$

где λ — интенсивность. Функция распределения экспоненциального закона

$$F(T) = \int_0^T p(t)dt = 1 - \exp(-\lambda T).$$

Из рис. 3.20 ясно, что поскольку искомыми являются значения b случайной величины T, то, задавая значение a, как равномерно распределенной в диапазоне [0,1] случайной величины, по формуле

$\beta = (1/\lambda)\ \ln(1/(1-\alpha))$ (3.47)

находим искомое значение. Именно в соответствии с (3.47) в операторах ADVANCE (см. пример 4) множителями были значения $1/\lambda$.

Приведем еще несколько операторов языка GPSS.

Оператор изменения параметров транзактов

```
ASSIGN A,B
```

где А — номер параметра транзакта, В — присваиваемое ему значение. В операторе

```
ASSIGN A+,B
```

параметр А увеличивается на значение В, а в операторе

```
ASSIGN A-,B
```

уменьшается. Расширение возможностей управления движением транзактов достигается благодаря таким операторам, как

```
LOGIC_X A
```

который при X = S устанавливает переключатель А в единичное состояние, а при X= R сбрасывает его в нулевое состояние;

```
GATE_XX A,B
```

который при XX = LR и A = 1 или при XX = LS и A = 0 передает транзакт оператору с меткой В (или задерживает его в блоке GATE, если поле В пусто), а при других сочетаниях XX и А — направляет к следующему оператору. Вычислительный оператор

```
M VARIABLE A
```

присваивает переменной с номером М значение арифметического выражения А, например в операторе

```
3 VARIABLE K216-S$MEM2
```

переменной номер 3 присваивается разность числа 216 и объема занятой памяти в накопителе MEM2. Оператор синхронизации, имеющий, например, вид

```
LBL MATCH NUMB
```

задерживает приходящий в него транзакт до тех пор, пока в некоторой другой части модели в сопряженный оператор

```
NUMB MATCH LBL
```

не войдет транзакт того же семейства.

Сети Петри. *Сети Петри* — аппарат для моделирования динамических дискретных систем (преимущественно асинхронных параллельных процессов). Сеть Петри определяется как четверка <**P**,**T**,**I**,**O**>, где **P** и **T** — конечные множества позиций и переходов, **I** и **O** — множества входных и выходных функций. Другими словами, сеть Петри представляет собой двудольный ориентированный граф, в котором *позициям* соответствуют вершины, изображаемые кружками, а *переходам* — вершины, изображаемые утолщенными черточками; функциям **I** соответствуют дуги, направленные от позиций к переходам, а функциям **O** — от переходов к позициям.

Как и в системах массового обслуживания, в сетях Петри вводятся объекты двух типов: динамические — изображаются *метками (маркерами)* внутри позиций и статические — им соответствуют вершины сети Петри.

Распределение маркеров по позициям называют *маркировкой*. Маркеры могут перемещаться в сети. Каждое изменение маркировки называют *событием*, причем каждое событие связано с определенным переходом. Считается, что события происходят мгновенно и разновременно при выполнении некоторых условий.

Каждому условию в сети Петри соответствует определенная позиция. Совершению события соответствует *срабатывание* (возбуждение или запуск) перехода, при котором маркеры из входных позиций этого перехода перемещаются в выходные позиции. Последовательность событий образует моделируемый процесс.

Правила срабатывания переходов (рис. 3.21), конкретизируют следующим образом: переход срабатывает, если для каждой из его входных позиций выполняется условие $N_i i\ K_i$, где N_i — число маркеров в i-й входной позиции, K_i — число дуг, идущих от i-й позиции к переходу; при срабатывании перехода число маркеров в i-й входной позиции уменьшается на K_i, а в j-й выходной позиции увеличивается на M_j, где M_j — число дуг, связывающих переход с j-й позицией.

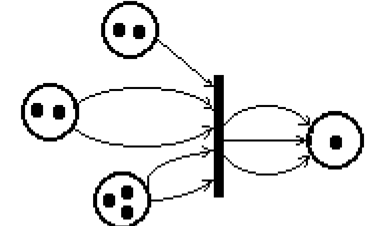


Рис.3.21. Фрагмент сети Петри

На рис. 3.21 показан пример распределения маркеров по позициям перед срабатыванием, эту маркировку записывают в виде (2,2,3,1). После срабатывания перехода маркировка становится иной: (1,0,1,4).

Можно вводить ряд дополнительных правил и условий в алгоритмы моделирования, получая ту или иную разновидность сетей Петри. Так, прежде всего полезно ввести модельное время, чтобы моделировать не только последовательность событий, но и их привязку ко времени. Это осуществляется приданием переходам веса — продолжительности (задержки) срабатывания, которую можно определять, используя задаваемый при этом алгоритм. Полученную модель называют *временной сетью Петри*.

Если задержки являются случайными величинами, то сеть называют *стохастической*. В стохастических сетях возможно введение вероятностей срабатывания возбужденных переходов. Так, на рис. 3.22 представлен фрагмент сети Петри, иллюстрирующий конфликтную ситуацию — маркер в позиции p может запустить либо переход t_1, либо переход t_2. В стохастической сети предусматривается вероятностный выбор срабатывающего перехода в таких ситуациях.

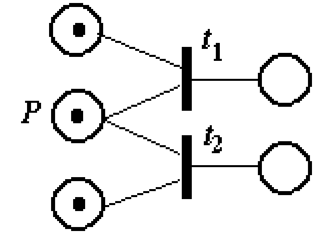


Рис.3.22. Конфликтная ситуация

Если задержки определяются как функции некоторых аргументов, которыми могут быть количества маркеров в каких-либо позициях, состояния некоторых переходов и т.п., то сеть называют *функциональной*.

Во многих задачах динамические объекты могут быть нескольких типов, и для каждого типа нужно вводить свои алгоритмы поведения в сети. В этом случае каждый маркер должен иметь хотя бы один параметр, обозначающий тип маркера. Такой параметр обычно называют цветом; цвет можно использовать как аргумент в функциональных сетях. Сеть Петри при этом называют *цветной*.

Среди других разновидностей сетей Петри следует упомянуть *ингибиторные* сети, характеризующиеся тем, что в них возможны запрещающие (ингибиторные) дуги. Наличие маркера во входной позиции, связанной с переходом ингибиторной дугой, означает запрещение срабатывания перехода.

Введенные понятия поясним на следующих примерах.

Пример 1. Требуется описать с помощью сети Петри работу группы пользователей на единственной рабочей станции WS при заданных характеристиках потока запросов на пользование WS и характеристиках поступающих задач. Сеть Петри представлена на рис. 3.23.

Здесь переходы связаны со следующими событиями: t_1 — поступление запроса на использование WS, t_2 — занятие станции, t_3 — освобождение станции, t_4 — выход обслуженной заявки; позиция p_4 используется для отображения состояния WS: если в p_4 имеется метка, то WS свободна и пришедшая заявка вызывает срабатывание перехода t_2; пока эта заявка не будет обслужена, метки в p_4 не будет, следовательно, пришедшие в позицию p_1 запросы вынуждены ожидать срабатывания перехода t_3.

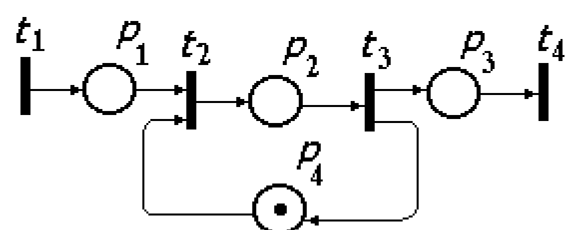


Рис. 3.23. Сеть Петри для примера 1

Пример 2. Требуется описать с помощью сети Петри процессы возникновения и устранения неисправностей в некоторой технической системе, состоящей из M однотипных блоков; в запасе имеется один исправный блок; известны статистические данные об интенсивностях возникновения отказов и длительностях таких операций, как поиск неисправ-

ностей, замена и ремонт отказавшего блока. На рис. 3.24 представлена соответствующая сеть Петри. Отметим, что при числе меток в позиции, равном M, можно в ней не ставить M точек, а записать в позиции значение M.

В нашем примере значение M в позиции p_2 соответствует числу имеющихся в системе блоков. Переходы отображают следующие события: t_1 — отказ блока, t_2 — поиск неисправного блока, t_3 — его замена, t_4 — окончание ремонта.

Очевидно, что при непустой позиции p_2 переход t_1 срабатывает, но с задержкой, равной вычисленному случайному значению моделируемого отрезка времени между отказами. После выхода маркера из t_1 он попадает через p_1 в t_2, если имеется метка в позиции p_6, это означает, что обслуживающая систему бригада специалистов свободна и может приступить к поиску возникшей неисправности. В переходе t_2 метка задерживается на время, равное случайному значению длительности поиска неисправности. Далее маркер оказывается в p_3 и, если имеется запасной блок (маркер в p_4), то запускается переход t_3, из которого маркеры выйдут в p_2, p_5 и в p_6 через отрезок времени, требуемый для замены блока. После этого в t_4 имитируется восстановление неисправного блока.

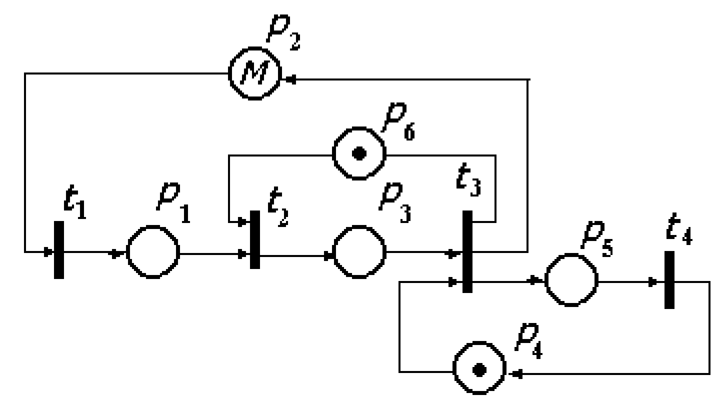


Рис. 3.24. Сеть Петри для примера 2

Рассматриваемая модель описывает функционирование системы в условиях, когда отказы могут возникать и в рабочем, и в неисправном состояниях системы. Поэтому не исключены ситуации, при которых более чем один маркер окажется в позиции p_1.

Анализ сетей Петри. Анализ сложных систем на базе сетей Петри можно выполнять посредством имитационного моделирования СМО, представленных моделями сетей Петри. При этом задают входные потоки заявок и определяют соответствующую реакцию системы. Выходные параметры СМО рассчитывают путем обработки накопленного при моделировании статистического материала. Возможен и другой подход к использованию сетей Петри для анализа объектов, исследуемых на системном уровне. Он не связан с имитацией процессов и основан на исследовании таких свойств сетей Петри, как ограниченность, безопасность, сохраняемость, достижимость, живость.

Ограниченность (или *K-ограниченность*) имеет место, если число меток в любой позиции сети не может превысить значения K. При проектировании автоматизированных систем определение K позволяет обоснованно выбирать емкости накопителей. Возможность неограниченного роста числа меток свидетельствует об опасности неограниченного роста длин очередей.

Безопасность — частный случай ограниченности, а именно это 1-ограниченность. Если для некоторой позиции установлено, что она безопасна, то ее можно представлять одним триггером.

Сохраняемость характеризуется постоянством загрузки ресурсов, т.е.

$$\sum A_i N_i = \text{const},$$

где N_i — число маркеров в i-й позиции, A_i — весовой коэффициент.

Достижимость $\mathbf{M}_k \rightarrow \mathbf{M}_j$ характеризуется возможностью достижения маркировки $\mathbf{M}_j$ из состояния сети, характеризуемого маркировкой $\mathbf{M}_k$.

Живость сети Петри определяется возможностью срабатывания любого перехода при функционировании моделируемого объекта. Отсутствие живости означает либо избыточность аппаратуры в проектируемой системе, либо свидетельствует о возможности возникновения зацикливаний, тупиков, блокировок.

В основе исследования перечисленных свойств сетей Петри лежит *анализ достижимости*.

Один из методов анализа достижимости любой маркировки из состояния $\mathbf{M}_o$ — построение графа достижимости. Начальная вершина графа отображает $\mathbf{M}_o$, а остальные вершины соответствуют маркировкам. Дуга из $\mathbf{M}_i$ в $\mathbf{M}_j$ означает событие $\mathbf{M}_i \rightarrow \mathbf{M}_j$ и соответствует срабатыванию перехода t. В сложных сетях граф может содержать чрезмерно большое число вершин и дуг. Однако при построении графа можно не отображать все вершины, так как многие из них являются дублями (действительно, от маркировки $\mathbf{M}_k$ всегда порождается один и тот же подграф вне зависимости от того. из какого состояния система пришла в $\mathbf{M}_k$). Тупики обнаруживаются по отсутствию разрешенных перехо-

дов из какой-либо вершины, т.е. по наличию листьев — терминальных вершин. Неограниченный рост числа маркеров в какой-либо позиции свидетельствует о нарушениях ограниченности.

Приведем примеры анализа достижимости.

Пример 1. Сеть Петри и граф достижимых разметок представлены на рис. 3.25.

На рисунке вершины графа изображены в виде маркировок, дуги помечены срабатывающими переходами. Живость сети очевидна, так как срабатывают все переходы, тупики отсутствуют.

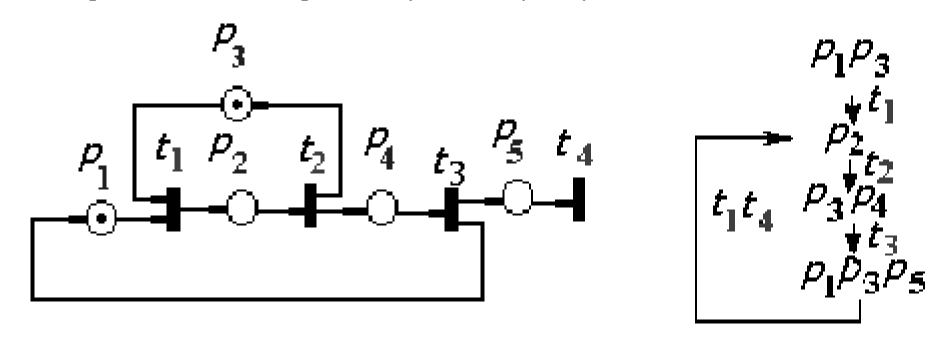


Рис. 3.25. Сеть Петри и ее граф достижимости для примера 1

Пример 2. Сеть Петри и граф достижимых разметок представлены на рис. 3.26.

Сеть, моделирующая двухпроцессорную вычислительную систему с общей памятью, является живой, все разметки достижимы.

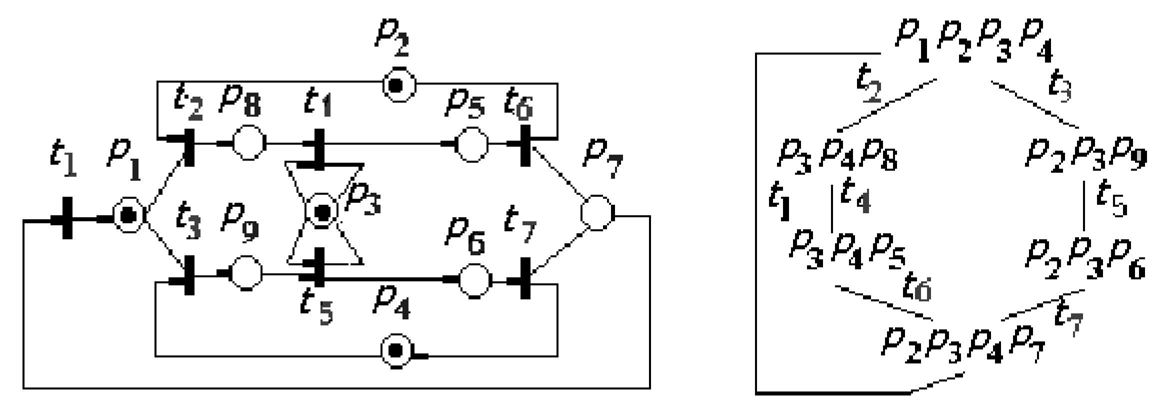


Рис. 3.26. Сеть Петри и ее граф достижимости для примера 2

3.7. Математическое обеспечение подсистем машинной графики и геометрического моделирования

Компоненты математического обеспечения. Подсистемы машинной графики и геометрического моделирования (МГиГМ) занимают центральное место в машиностроительных САПР-К. Конструирование изделий в них, как правило, проводится в интерактивном режиме при оперировании геометрическими моделями, т.е. математическими объектами, отображающими форму деталей, состав сборочных узлов и возможно некоторые дополнительные параметры (масса, момент инерции, цвета поверхности и т.п.).

В подсистемах МГиГМ типичный маршрут обработки данных включает в себя получение проектного решения в прикладной программе, его представление в виде геометрической модели (геометрическое моделирование), подготовку проектного решения к визуализации, собственно визуализацию в аппаратуре рабочей станции и при необходимости корректировку решения в интерактивном режиме. Две последние операции реализуются на базе аппаратных средств машинной графики. Когда говорят о математическом обеспечении МГиГМ, имеют в виду прежде всего модели, методы и алгоритмы для геометрического моделирования и подготовки к визуализации. При этом часто именно математическое обеспечение подготовки к визуализации называют математическим обеспечением машинной графики.

Различают математическое обеспечение двумерного (2*D*) и трехмерного (3*D*) моделирования. Основные применения 2*D* графики — подготовка чертежной документации в машиностроительных САПР, топологическое проектирование печатных плат и кристаллов БИС в САПР электронной про-

мышленности. В развитых машиностроительных САПР используют как 2*D*, так и 3*D* моделирование для синтеза конструкций, представления траекторий рабочих органов станков при обработке заготовок, генерации сетки конечных элементов при анализе прочности и т.п.

В 3*D* моделировании различают модели каркасные (проволочные), поверхностные, объемные (твердотельные).

Каркасная модель представляет форму детали в виде конечного множества линий, лежащих на поверхностях детали. Для каждой линии известны координаты концевых точек и указана их инцидентность ребрам или поверхностям. Оперировать каркасной моделью на дальнейших операциях маршрутов проектирования неудобно, и поэтому каркасные модели в настоящее время используют редко.

Поверхностная модель отображает форму детали с помощью задания ограничивающих ее поверхностей, например, в виде совокупности данных о гранях, ребрах и вершинах.

Особое место занимают модели деталей с поверхностями сложной формы, так называемыми *скульптурными поверхностями*. К таким деталям относятся корпуса многих транспортных средств (например, судов, автомобилей), детали, обтекаемые потоками жидкостей и газов (лопатки турбин, крылья самолетов), и др.

Объемные модели отличаются тем, что в них в явной форме содержатся сведения о принадлежности элементов внутреннему или внешнему по отношению к детали пространству.

В настоящее время применяют следующие подходы к построению геометрических моделей.

1. Задание граничных элементов — граней, ребер, вершин.

2. *Кинематический метод*, согласно которому задают двумерный контур и траекторию его перемещения; след от перемещения контура принимают в качестве поверхности детали.

3. *Позиционный подход*, в соответствии с которым рассматриваемое пространство разбивают на ячейки (позиции) и деталь задают указанием ячеек, принадлежащих детали; очевидна громоздкость этого подхода.

4. Представление сложной детали в виде совокупностей *базовых элементов формы* (БЭФ) и выполняемых над ними теоретико-множественных операций. К БЭФ относятся заранее разработанные модели простых тел, это, в первую очередь, модели параллелепипеда, цилиндра, сферы, призмы. Типичными теоретико-множественными операциями являются объединение, пересечение, разность. Например, модель плиты с отверстием в ней может быть получена вычитанием цилиндра из параллелепипеда.

Метод на основе БЭФ часто называют *методом конструктивной геометрии*. Это основной способ конструирования сборочных узлов в современных САПР-К.

В памяти ЭВМ рассмотренные модели обычно хранятся в векторной форме, т.е. в виде координат совокупности точек, задающих элементы модели. Выполнение операций конструирования также выполняется над моделями в векторной форме. Наиболее компактна модель в виде совокупности связанных БЭФ, которая преимущественно и используется для хранения и обработки информации об изделиях в системах конструктивной геометрии.

Однако для визуализации в современных рабочих станциях в связи с использованием в них растровых дисплеев необходима *растеризация* — преобразование модели в растровую форму. Обратную операцию перехода к векторной форме, которая характеризуется меньшими затратами памяти, называют *векторизацией*. В частности, векторизация должна выполняться по отношению к данным, получаемым сканированием изображений в устройствах автоматического ввода.

Геометрические модели. Важной составной частью геометрических моделей является описание поверхностей. Если поверхности детали — плоские грани, то модель может быть выражена достаточно просто определенной информацией о гранях, ребрах, вершинах детали. При этом обычно используется метод конструктивной геометрии. Представление с помощью плоских граней имеет место и в случае более сложных поверхностей, если эти поверхности аппроксимировать множествами плоских участков — полигональными сетками. Тогда можно поверхностную модель задать одной из следующих форм:

1) модель есть список граней, каждая грань представлена упорядоченным списком вершин (циклом вершин); эта форма характеризуется значительной избыточностью, так как каждая вершина по-

вторяется в нескольких списках.

2) модель есть список ребер, для каждого ребра заданы инцидентные вершины и грани.

Однако аппроксимация полигональными сетками при больших размерах ячеек сетки дает заметные искажения формы, а при малых размерах ячеек оказывается неэффективной по вычислительным затратам. Поэтому более популярны описания неплоских поверхностей кубическими уравнениями в форме Безье или *B*-сплайнов.

Знакомство с этими формами удобно выполнить, показав их применение для описания геометрических объектов первого уровня — пространственных кривых.

Примечание. Геометрическими объектами нулевого, первого и второго уровней называют соответственно точки, кривые, поверхности.

В подсистемах МГиГМ используются параметрически задаваемые кубические кривые

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x ,$$
$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y , \qquad (3.48)$$
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z ,$$

где 1 i *t* i 0. Такими кривыми описывают сегменты аппроксимируемой кривой, т.е. аппроксимируемую кривую разбивают на сегменты и каждый сегмент аппроксимируют уравнениями (3.48).

Применение кубических кривых обеспечивает (соответствующим выбором четырех коэффициентов в каждом из трех уравнений) выполнение четырех условий сопряжения сегментов. В случае кривых Безье этими условиями являются прохождение кривой сегмента через две заданные концевые точки и равенство в этих точках касательных векторов соседних сегментов. В случае *B*-сплайнов выполняются условия непрерывности касательного вектора и кривизны (т.е. первой и второй производных) в двух концевых точках, что обеспечивает высокую степень “гладкости” кривой, хотя прохождение аппроксимирующей кривой через заданные точки здесь не обеспечивается. Применение полиномов выше третьей степени не рекомендуется, так как велика вероятность появления “волнистости”.

В случае формы Безье коэффициенты в (3.48) определяются, во-первых, подстановкой в (3.48) значений t=0 и t=1 и координат заданных концевых точек $\mathbf{P}_1$ и $\mathbf{P}_4$ соответственно, во-вторых, подстановкой в выражения производных

$$dx/dt = 3a_x t^2 + 2b_x + c_x ,$$
$$dy/dt = 3a_y t^2 + 2b_y t + c_y ,$$
$$dz/dt = 3a_z t^2 + 2b_z t + c_z ,$$

тех же значений t=0 и t=1 и координат точек $\mathbf{P}_2$ и $\mathbf{P}_3$, задающих направления касательных векторов (рис. 3.27). В результате для формы Безье получаем

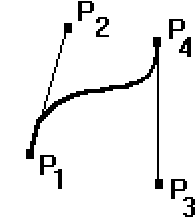


Рис. 3.27.Кривая Безье

$$x(t) = \mathbf{T}^{\mathrm{T}}\mathbf{M}\mathbf{G}_x,$$
$$y(t) = \mathbf{T}^{\mathrm{T}}\mathbf{M}\mathbf{G}_y, \qquad (3.49)$$
$$z(t) = \mathbf{T}^{\mathrm{T}}\mathbf{M}\mathbf{G}_z,$$

Таблица 3.11

-1	3	-3	1
3	-6	3	0
-3	3	0	0
1	0	0	0

где $\mathbf{T}^{\mathrm{T}} = (\, t^3,\ t^2,\ t,\ 1)$ — вектор-строка, матрица $\mathbf{M}$ представлена в табл. 3.11, $\mathbf{G}_x$ — вектор координат P_{xi} точек $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$ и $\mathbf{P}_4$, аналогично $\mathbf{G}_y$, $\mathbf{G}_z$ _ векторы координат P_{yi}, P_{zi} тех же точек.

В случае *B*-сплайнов аппроксимируемая кривая делится на *n* участков, выделяемых последовательными точками $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$,…$\mathbf{P}_n$. Участок между парой соседних точек $\mathbf{P}_i$ и $\mathbf{P}_{i+1}$ аппроксимируется *B*-сплайном, построенным с использованием четырех точек $\mathbf{P}_{i-1}$, $\mathbf{P}_i$, $\mathbf{P}_{i+1}$, $\mathbf{P}_{i+2}$. *B*-сплайн на участке $[\mathbf{P}_i,\ \mathbf{P}_{i+1}]$ может быть представлен выражениями, аналогичными (3.49),

$$x(t) = \mathbf{T}^{\mathrm{T}}\mathbf{M}\mathbf{G}_x,$$
$$y(t) = \mathbf{T}^{\mathrm{T}}\mathbf{M}\mathbf{G}_y,$$
$$z(t) = \mathbf{T}^{\mathrm{T}}\mathbf{M}\mathbf{G}_z,$$

Таблица 3.12

-1/6	1/2	-1/2	1/6
1/2	-1	1/2	0
-1/2	0	1/2	0
1/6	2/3	1/6	0

для которых матрица **M** имеет иной вид и представлена в табл. 3.12, а векторы $\mathbf{G}_x$, $\mathbf{G}_y$, $\mathbf{G}_z$ содержат соответствующие координаты точек $\mathbf{P}_{i-1}$, $\mathbf{P}_i$, $\mathbf{P}_{i+1}$, $\mathbf{P}_{i+2}$.

Покажем, что в точках сопряжения для первой и второй производных аппроксимирующего выражения выполняются условия непрерывности, что требуется по определению *B*-сплайна. Обозначим участок аппроксимирующего B-сплайна, соответствующий участку $[\mathbf{P}_i, \mathbf{P}_{i+1}]$ исходной кривой, через $[\mathbf{Q}_i, \mathbf{Q}_{i+1}]$. Тогда для этого участка и координаты x в точке сопряжения $\mathbf{Q}_{i+1}$ имеем t=1 и

$$dx(t)/dt|_{t=1} = [3t^2,2t,1,0]*\mathbf{M}*[x_{i-1},x_i,x_{i+1},x_{i+2}]^T = [3,2,1,0]*\mathbf{M}*[x_{i-1},x_i,x_{i+1},x_{i+2}]^T = (x_{i+2}-x_i) / 2;$$
$$d^2x(t)/dt^2|_{t=1} = [6t,2,0,0]*\mathbf{M}*[x_{i-1},x_i,x_{i+1},x_{i+2}]^T = [6,2,0,0]*\mathbf{M}*[x_{i-1},x_i,x_{i+1},x_{i+2}]^T = x_i-2\,x_{i+1}+x_{i+2}.$$

Для участка $[\mathbf{Q}_{i+1}, \mathbf{Q}_{i+2}]$ в той же точке $\mathbf{Q}_{i+1}$ имеем t=0 и

$$dx(t)/dt|_{t=0} = [0,0,1,0]*\mathbf{M}*[x_i, x_{i+1},x_{i+2}, x_{i+3}]^T = (x_{i+2}-x_i) / 2,$$
$$d^2x(t)/dt^2|_{t=0} = [0,2,0,0]*\mathbf{M}*[x_i, x_{i+1},x_{i+2}, x_{i+3}]^T = x_i-2\,x_{i+1}+x_{i+2},$$

т.е. равенство производных в точке сопряжения на соседних участках подтверждает непрерывность касательного вектора и кривизны. Естественно, что значение x_q координаты x точки $\mathbf{Q}_{i+1}$ аппроксимирующей кривой на участке $[\mathbf{Q}_i, \mathbf{Q}_{i+1}]$

$$x_q = [1,1,1,1]*M*[x_{i-1},x_i,x_{i+1},x_{i+2}]^T = (x_i+4\,x_{i+1}+x_{i+2}) / 6$$

равно значению x_q, подсчитанному для той же точки на участке $[\mathbf{Q}_{i+1}, \mathbf{Q}_{i+2}]$, но значения координат узловых точек x_q и x_{i+1} аппроксимирующей и аппроксимируемой кривых не совпадают.

Аналогично можно получить выражения для форм Безье и *B*-сплайнов применительно к поверхностям с учетом того, что вместо (3.48) используются кубические зависимости от двух переменных.

Методы и алгоритмы машинной графики (подготовки к визуализации). К методам машинной графики относят методы преобразования графических объектов, представления (развертки) линий в растровой форме, выделения окна, удаления скрытых линий, проецирования, закраски изображений.

Преобразования графических объектов выполняются с помощью операций переноса, масштабирования, поворота.

Перенос точки из положения **P** в новое положение **C** можно выполнять по формулам типа

$$C_{xi} = P_{xi} + \Delta x_i,$$

где Δx_i — приращение по координате x_i. Однако удобнее операции преобразования представлять в единой матричной форме

$$\mathbf{C} = \mathbf{P}\ \mathbf{T}, \qquad (3.50)$$

где **T** — преобразующая матрица. При этом точки **C** и **P** в двумерном случае изображают векторами-строками 1×3, в которых кроме значений двух координат, называемых при таком представлении однородными, дополнительно указывают масштабный множитель W. Тогда перенос для случая 2*D* можно выразить в виде (3.50), где **T** есть табл. 3.13, а $W = 1$.

Для операций масштабирования и поворота матрицы **T** представлены в табл. 3.14 и 3.15 соответственно, где m_x, m_y — масштабные множители, φ — угол поворота.

Таблица 3.13

1	0	0
0	1	0
Δx_1	Δx_2	1

Таблица 3.14

m_x	0	0
0	m_y	0
0	0	1

Таблица 3.15

cos φ	sin φ	0
-sin φ	cos φ	0
0	0	1

Удобство (3.50) объясняется тем, что любую комбинацию элементарных преобразований можно описать формулой (3.50). Например, выражение для сдвига с одновременным поворотом имеет вид

$$\mathbf{C} = \mathbf{P}\ \mathbf{T}_{сд}\ \mathbf{T}_{пов} = \mathbf{P}\ \mathbf{T},$$

где $\mathbf{T} = \mathbf{T}_{сд}\ \mathbf{T}_{пов}$, $\mathbf{T}_{сд}$ — матрица сдвига, $\mathbf{T}_{пов}$ — матрица поворота.

Представление графических элементов в растровой форме требуется для отображения этих элементов на битовую карту растровой видеосистемы. Пусть требуется развернуть отрезок **AB** прямой $y = ax+b$, причем $1 \geq a \geq 0$. Введем обозначения: $\mathbf{A} = (xa, ya)$, $\mathbf{B} = (xb, yb)$; за величину дискрета (пик-

села) примем единицу. В алгоритме развертки номера строк и столбцов карты, на пересечении которых должны находиться точки отрезка, определяются следующим образом:

1) $\Delta x := xb - xa$; $\Delta y := yb - ya$; $x := xa$; $y := ya$;

2) $d = 2\,\Delta y - \Delta x$;

3) если $d \geq 0$, то $\{y := y+1;\ d := d + 2(\Delta y-\Delta x)\}$; иначе $d := d + 2\,\Delta y$;

4) $x := x + 1$;

5) переход к пункту 3, пока не достигнута точка **B.**

Этот же алгоритм с небольшими модификациями используется и при других значениях коэффициента a.

Экономичность этого алгоритма обусловливается отсутствием длинных арифметических операций типа умножения.

Выделение окна требуется при определении той части сцены, которая должна быть выведена на экран дисплея.

Пусть окно ограничено линиями $x = x_1$, $x = x_2$, $y = y_1$, $y = y_2$ (рис. 3.28). Поочередно для каждого многоугольника проверяется расположение его вершин и ребер относительно границ окна. Так, для многоугольника **ABCD** (см. рис. 3.28) при отсечении по границе $x = x_2$ просматривается множество вершин в порядке обхода по часовой стрелке. Возможны четыре ситуации для двух последовательных вершин **P** и **R**:

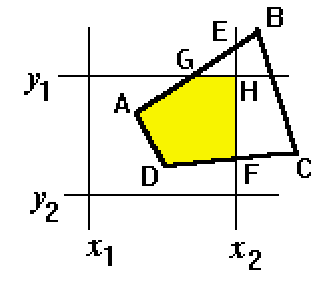


Рис. 3.28. Выделение окна

1) если $x_p > x_2$ и $x_R > x_2$, то обе вершины и инцидентное им ребро находятся вне окна и исключаются из дальнейшего анализа;

2) если $x_p \leq x_2$ и $x_R \leq x_2$, то обе вершины и инцидентное им ребро остаются для дальнейшего анализа;

3) если $x_p \leq x_2$ и $x_R > x_2$, то вершина **P** остается в списке вершин, а вершина **R** заменяется новой вершиной с координатами $x = x_2$, $y = y_p + (y_R-y_p)(x_2-x_P)/(x_R-x_P)$; в нашем примере такой новой вершиной будет **E;**

4) если $x_p > x_2$ и $x_R \leq x_2$, то вершина **P** заменяется новой вершиной с координатами $x = x_2$, $y = y_R + (y_P-y_R)(x_2-x_R)/(x_P-x_R)$, а вершина **R** остается в списке вершин; в нашем примере новой вершиной будет **F.**

После окончания просмотра применительно ко всем границам в окне оказываются оставшиеся в списке вершины.

Применяя эти правила в нашем примере, получаем сначала многоугольник **AEFD**, а после отсечения по верхней границе $y = y_2$ — многоугольник **AGFD** (см. рис.3.28). Однако правильный результат несколько иной, а именно многоугольник **AGHFD**. Этот правильный результат получается при двойном обходе вершин сначала по часовой стрелке, затем против с включением в список новых вершин, появляющихся при каждом обходе.

Применяют ряд алгоритмов *удаления скрытых линий*. Один из наиболее просто реализуемых алгоритмов — алгоритм z-буфера, где z-буфер — область памяти, число ячеек в которой равно числу пикселов в окне вывода. Предполагается, что ось z направлена по нормали к видовой поверхности и наблюдатель расположен в точке $z = 0$.

В начале исполнения алгоритма все пикселы соответствуют максимальному значению z, т.е. максимальному удалению от наблюдателя, что приводит к помещению во все ячейки z-буфера значений пикселов фона картины (чертежа). Далее поочередно для всех точек граней рассчитываются значения координаты z. Среди точек, относящихся к одному и тому же пикселу (одной и той же ячейке z-буфера S), выбирается точка с наименьшим значением z и ее код (т.е. цвет и яркость) помещается в S. В итоге z-буфер будет содержать пикселы наиболее близких к наблюдателю граней.

Алгоритмы построения проекций преобразуют трехмерные изображения в двумерные. В случае построения центральной проекции каждая точка трехмерного изображения отображается на картинную поверхность путем пересчета координат x и y (рис. 3.29). Так, координату x_a’ точки **A’** вычисля-

ют по очевидной формуле

$$x_a' = x_a\, d/z,$$

аналогично рассчитывается координата y_a' точки **А'**.

В параллельных проекциях $d \to \infty$ и координаты x и y точек **А'** и **А** совпадают. Поэтому построение параллельных проекций сводится к выделению окна, при необходимости к повороту изображения и возможно к удалению скрытых линий.

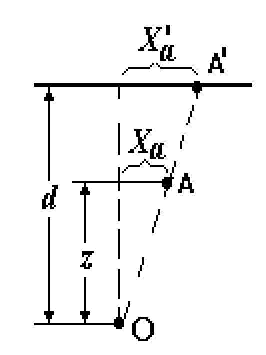


Рис. 3.29. Построение центральной проекции точки А

Закраска матовых поверхностей основана на законе Ламберта, согласно которому яркость отраженного от поверхности света пропорциональна cosa, где a — угол между нормалью к поверхности и направлением луча падающего света. В алгоритме Гуро яркость внутренних точек определяется линейной интерполяцией яркости в вершинах многоугольника. При этом сначала проводится интерполяция в точках ребер, а затем по строкам горизонтальной развертки. Более реалистичными получаются изображения в алгоритме Фонга, основанном на линейной интерполяции векторов нормалей к поверхности.

Программно-методические комплексы геометрического моделирования и машинной графики. Мировыми лидерами в этой области программного обеспечения САПР считаются Pro/Engineer, Unigraphics, EUCLID, CADDS5, CATIA, I-DEAS и ряд других. Отметим, что по своим функциональным возможностям эти комплексы приблизительно равноценны, новые достижения, появившиеся в одном из них, в скором времени реализуются в новых версиях других комплексов. Поэтому для первого знакомства достаточно рассмотреть характеристики одного из комплексов. Ниже приведены краткие сведения по комплексу Pro/Engineer.

Комплекс насчитывает несколько десятков программ (модулей), которые разделены на группы программ конструкторского проектирования механических объектов, промышленного дизайна, функционального моделирования, технологического проектирования, обмена данными.

Базовые модули конструкторского проектирования предназначены для твердотельного и поверхностного моделирования, синтеза конструкций из базовых элементов формы, поддерживают параметризацию и ассоциативность, проекционное черчение, выполняют разработку чертежей с простановкой размеров и допусков. Пользователь может пополнять библиотеку БЭФ оригинальными моделями. Синтез трехмерных моделей сложной формы возможен вытягиванием плоского контура по нормали к его плоскости, его протягиванием вдоль произвольной пространственной кривой, вращением контура вокруг заданной оси, натягиванием между несколькими заданными сечениями. Синтез сборок выполняется вызовом или ссылкой на библиотечные элементы, их модификацией, разработкой новых деталей. Детали сборки можно нужным образом ориентировать в пространстве. Далее следует ввести ассоциативные (сопрягающие) связи.

Дополнительные модули конструкторского проектирования имеют более конкретную, но узкую специализацию. Примерами таких модулей могут служить модули конструирования панелей из композитных материалов, разработки штампов и литейных пресс-форм, трубопроводных систем, сварных конструкций, разводки электрических кабелей и жгутов.

Модули функционального моделирования используются как препроцессоры и постпроцессоры для программ конечно-элементного анализа (нанесение сетки конечных элементов, визуализация результатов анализа), для анализа теплового состояния конструкций, для оценки виброустойчивости и др.

Основные модули технологического проектирования служат для моделирования технологических процессов фрезерной, токарной, электроэрозионной обработки и для разработки постпроцессоров для систем управления оборудованием с ЧПУ.

Среди модулей обмена данными важно наличие взаимосвязей по стандарту STEP, что открывает возможности импорта/экспорта данных с различными CAE/CAD/CAM системами, поддерживающими этот стандарт.

Упражнения и вопросы для самоконтроля

1. Дайте определение области адекватности математической модели.

2. Представьте схему рис. 3.30 в виде графа, постройте покрывающее дерево, запишите матрицу контуров и сечений **M**.

3. Запишите компонентные и топологические уравнения для эквивалентной схемы рис. 3.30.

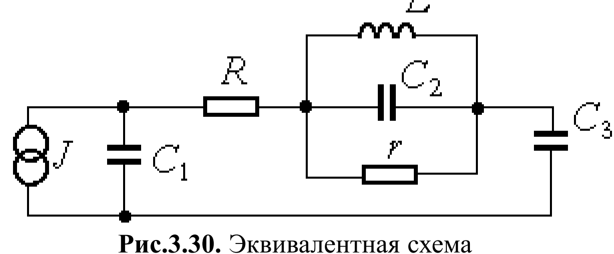


Рис.3.30. Эквивалентная схема

4. Составьте эквивалентную схему для гидромеханической системы (цилиндра с поршнем), представленную на рис. 3.31, где *F* — сила, действующая на поршень.

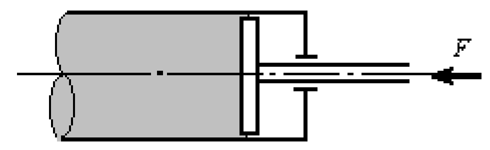
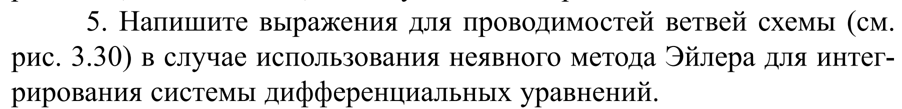
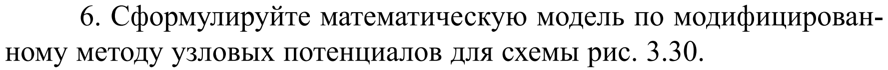
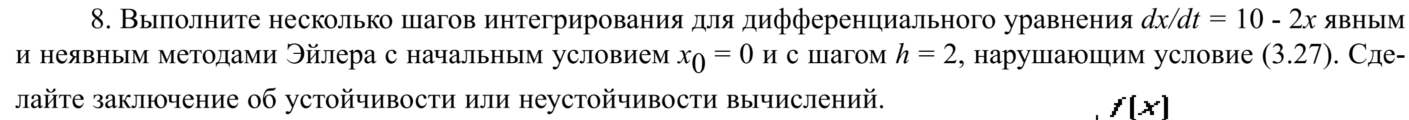


Рис.3.31. Гидромеханическая система


5. Напишите выражения для проводимостей ветвей схемы (см. рис. 3.30) в случае использования неявного метода Эйлера для интегрирования системы дифференциальных уравнений.


6. Сформулируйте математическую модель по модифицированному методу узловых потенциалов для схемы рис. 3.30.


7. Что понимают под постоянной времени физической системы?


8. Выполните несколько шагов интегрирования для дифференциального уравнения $dx/dt = 10 - 2x$ явным и неявным методами Эйлера с начальным условием $x_0 = 0$ и с шагом $h = 2$, нарушающим условие (3.27). Сделайте заключение об устойчивости или неустойчивости вычислений.

9. Каким образом обеспечивается сходимость итераций при решении СНАУ?

10. На чем основаны алгоритмы автоматического выбора шага интегрирования при решении систем дифференциальных уравнений?

11. Что такое “вторичные ненулевые элементы” в методах разреженных матриц?

12. В чем заключается различие способов интерпретации и компиляции при реализации метода разреженных матриц?

13. Что понимают под областью работоспособности?

14. Найдите координатные функции для одномерной задачи при линейной аппроксимации функции $f(x)$ (рис. 3.32, на котором показаны конечные элементы длиной L).

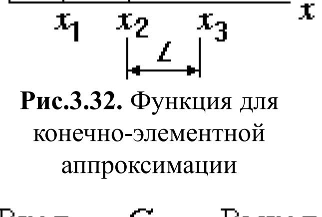


Рис.3.32. Функция для конечно-элементной аппроксимации

15. Найдите передаточную функцию для схемы рис. 3.33.

16. Постройте таблицы логических функций И и ИЛИ для пятизначного алфавита.

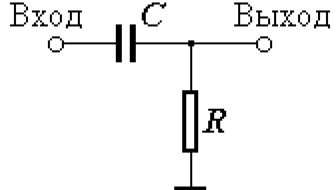


Рис.3.33. Дифференцирующая цепь

17. Поясните сущность событийного метода моделирования.

18. Приведите вывод уравнений Колмогорова для систем массового обслуживания.

19. Постройте граф состояний для системы массового обслуживания, состоящей из двух идентичных обслуживающих аппаратов (ОА) с интенсивностью обслуживания μ каждый и включенных параллельно при общем входном потоке с интенсивностью поступления заявок λ. Если свободны оба ОА, пришедшая заявка занимает первый ОА. Если очередь равна 2, то приходящие заявки покидают систему без обслуживания.

20. Опишите на языке GPSS модель системы, состоящей из трех станков и обрабатывающей детали типов *A* и *B*. Заданы интенсивности поступления деталей этих типов и интенсивности обработки их на каждом станке. Маршруты деталей типа *A* включают станки 1 и 2, деталей типа *B* — станки 1 и 3.

21. Как и в предыдущем примере на входе системы имеются потоки деталей типов *A* и *B*, но система представляет собой сборочную линию, на выходе которой каждое изделие состоит из *n* деталей типа *A* и *m* деталей типа *B*. Требуется разработать модель системы и представить ее на языке GPSS.

22. Запишите на языке GPSS модель системы, представленной на рис. 3.24 в виде сети Петри.

23. Что такое “параметрическая модель”и “ассоциативное моделирование”?

24. Представьте матрицу преобразования, включающего сжатие плоского изображения в *k* раз и его перемещение вдоль оси *x* на величину *D*.

25. В чем заключаются отличия геометрических моделей Безье и B-сплайнов?



Математическое обеспечение синтеза проектных решений

4.1. Постановка задач параметрического синтеза

Место процедур синтеза в проектировании. Сущность проектирования заключается в принятии проектных решений, обеспечивающих выполнение будущим объектом предъявляемых к нему требований. Синтез проектных решений — основа проектирования; от успешного выполнения процедуры синтеза в определяющей мере зависят потребительские свойства будущей продукции. Конечно, анализ — необходимая составная часть проектирования, служащая для верификации принимаемых проектных решений. Именно анализ позволяет получить необходимую информацию для целенаправленного выполнения процедур синтеза в итерационном процессе проектирования. Поэтому синтез и анализ неразрывно связаны.

Как отмечено в гл. 1, синтез подразделяют на параметрический и структурный. Проектирование начинается со *структурного синтеза*, при котором генерируется принципиальное решение. Таким решением может быть облик будущего летательного аппарата, или физический принцип действия датчика, или одна из типовых конструкций двигателя, или функциональная схема микропроцессора. Но эти конструкции и схемы выбирают в параметрическом виде, т.е. без указания числовых значений параметров элементов. Поэтому прежде чем приступить к верификации проектного решения, нужно задать или рассчитать значения этих параметров, т.е. выполнить *параметрический синтез*. Примерами результатов параметрического синтеза могут служить геометрические размеры деталей в механическом узле или в оптическом приборе, параметры электрорадиоэлементов в электронной схеме, параметры режимов резания в технологической операции и т.п.

В случае если по результатам анализа проектное решение признается неокончательным, то начинается процесс последовательных приближений к приемлемому варианту проекта. Во многих приложениях для улучшения проекта удобнее варьировать значения параметров элементов, т.е. использовать параметрический синтез на базе многовариантного анализа. При этом задача параметрического синтеза может быть сформулирована как задача определения значений параметров элементов, наилучших с позиций удовлетворения требований технического задания при неизменной структуре проектируемого объекта. Тогда параметрический синтез называют параметрической оптимизацией или просто *оптимизацией*. Если параметрический синтез не приводит к успеху, то повторяют процедуры структурного синтеза, т.е. на очередных итерациях корректируют или перевыбирают структуру объекта.

Критерии оптимальности. В САПР процедуры параметрического синтеза выполняются либо человеком в процессе многовариантного анализа (в интерактивном режиме), либо реализуются на базе формальных методов оптимизации (в автоматическом режиме). В последнем случае находят применение несколько постановок задач оптимизации.

Наиболее распространенной является детерминированная постановка: заданы условия работоспособности на выходные параметры $\mathbf{Y}$ и нужно найти номинальные значения проектных параметров $\mathbf{X}$, к которым относятся параметры всех или части элементов проектируемого объекта. Назовем эту задачу оптимизации базовой. В частном случае, когда требования к выходным параметрам заданы нечетко, к числу рассчитываемых величин могут быть отнесены также нормы выходных параметров, фигурирующие в их условиях работоспособности.

Если проектируются изделия для дальнейшего серийного производства, то важное значение приобретает такой показатель, как процент выпуска годных изделий в процессе производства. Очевидно, что успешное выполнение условий работоспособности в номинальном режиме не гарантирует их выполнения при учете производственных погрешностей, задаваемых допусками параметров элементов. Поэтому *целью оптимизации становится максимизация процента выхода годных*, а к результатам решения задачи оптимизации относятся не только номинальные значения проектных параметров, но и их допуски.

Базовая задача оптимизации ставится как задача математического программирования:

$$\underset{\mathbf{X}\in\mathbf{D}_x}{\mathbf{extr}}\, F(\mathbf{X}), \qquad (4.1)$$

$$\mathbf{D}_x = \{\mathbf{X} |\ \boldsymbol{\varphi}(\mathbf{X}) > 0,\ \boldsymbol{\psi}(\mathbf{X}) = 0\},$$

где $F(\mathbf{X})$ — целевая функция, $\mathbf{X}$ — вектор управляемых (проектных) параметров, $\varphi(\mathbf{X})$ и $\psi(\mathbf{X})$ — функции-ограничения, $\mathbf{D}_x$ — допустимая область в пространстве управляемых параметров. Запись (4.1) интерпретируется как задача поиска экстремума целевой функции путем варьирования управляемых параметров в пределах допустимой области.

Таким образом, для выполнения расчета номинальных значений параметров необходимо, во-первых, сформулировать задачу в виде (4.1), во-вторых, решить задачу поиска экстремума $F(\mathbf{X})$.

Сложность постановки оптимизационных проектных задач обусловлена наличием у проектируемых объектов нескольких выходных параметров, которые могут быть критериями оптимальности, но в задаче (4.1) целевая функция должна быть одна. Другими словами, проектные задачи являются многокритериальными, и возникает проблема сведения многокритериальной задачи к однокритериальной.

Применяют несколько способов выбора критерия оптимальности.

В *частном критерии* среди выходных параметров один выбирают в качестве целевой функции, а условия работоспособности остальных выходных параметров относят к ограничениям задачи (4.1). Эта постановка вполне приемлема, если действительно можно выделить один наиболее критичный выходной параметр. Но в большинстве случаев сказывается недостаток частного критерия (рис. 4.1).

На этом рисунке представлено двумерное пространство выходных параметров y_1 и y_2, для которых заданы условия работоспособности $y_1 < T_1$ и $y_2 < T_2$. Кривая **AB** является границей достижимых значений выходных параметров. Это ограничение объективное и связано с существующими физическими и технологическими условиями производства, называемыми условиями реализуемости. Область, в пределах которой выполняются все условия реализуемости и работоспособности, называют *областью работоспособности*. Множество точек пространства выходных параметров, из которых невозможно перемещение, приводящее к улучшению всех выходных параметров, называют областью компромиссов, или *областью Парето*. Участок кривой **AB** (см. рис. 4.1) относится к области Парето.

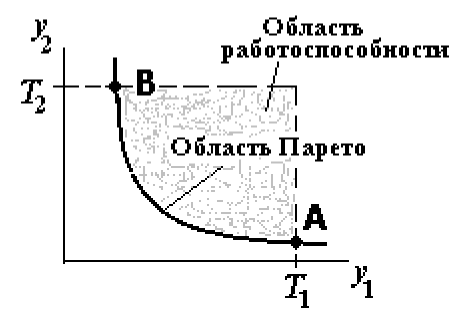


Рис. 4.1. Области Парето и работоспособности

Если в качестве целевой функции в ситуации рис. 4.1. выбрать параметр y_1, то результатом оптимизации будут параметры **X**, соответствующие точке **B**. Но это граница области работоспособности и, следовательно, при нестабильности внутренних и внешних параметров велика вероятность выхода за пределы области работоспособности. Конечно, результаты можно улучшить, если применять так называемый метод уступок, при котором в качестве ограничения принимают условие работоспособности со скорректированной нормой в виде

$$y_2 < T_2 + \Delta,$$

где Δ — уступка. Но возникает проблема выбора значений уступок, т.е. результаты оптимизации будут иметь субъективный характер. Очевидно, что ситуация не изменится, если целевой функцией будет выбран параметр y_2, — оптимизация приведет в точку **A** .

Аддитивный критерий объединяет (свертывает) все выходные параметры (частные критерии) в одну целевую функцию, представляющую собой взвешенную сумму частных критериев

$$F(\mathbf{X}) = \sum_{j=1}^{m} \omega_j \, y_j(\mathbf{X}), \qquad (4.2)$$

где ω_j — весовой коэффициент, m — число выходных параметров. Функция (4.2) подлежит минимизации, при этом если условие работоспособности имеет вид $y_j > T_j$, то $\omega_j < 0$.

Недостатки аддитивного критерия — субъективный подход к выбору весовых коэффициентов и неучет требований ТЗ. Действительно в (4.2) не входят нормы выходных параметров.

Аналогичные недостатки присущи и *мультипликативному критерию*, целевая функция которого имеет вид

$$F(\mathbf{X}) = \prod_{j=1}^{m} y_j^{\omega_j}(\mathbf{X}). \qquad (4.3)$$

Нетрудно видеть, что если прологарифмировать (4.3), то мультипликативный критерий превращается в аддитивный.

Более предпочтительным является *максиминный критерий*, в качестве целевой функции которого принимают выходной параметр, наиболее неблагополучный с позиций выполнения условий работоспособности. Для оценки степени выполнения условия работоспособности j-го выходного параметра вводят запас работоспособности этого параметра S_j и этот запас можно рассматривать как нормированный j-й выходной параметр. Например (здесь и далее для лаконичности изложения предполагается, что все выходные параметры приведены к виду, при котором условия работоспособности становятся неравенствами в форме $y_j < T_j$):

$$S_j = (T_j - y_j) / T_j$$

или

$$S_j = (T_j - y_{\text{ном}\, j}) / \delta_j ,$$

где $y_{\text{ном}\, j}$ — номинальное значение, а δ_j — некоторая характеристика рассеяния j-го выходного параметра, например, трехсигмовый допуск. Тогда целевая функция в максиминном критерии есть

$$F(\mathbf{X}) = \min_{j \in [1:m]} Z_j(\mathbf{X}).$$

Здесь запись [1: m] означает множество целых чисел в диапазоне от 1 до m. Задача (4.1) при максиминном критерии конкретизируется следующим образом:

$$F(\mathbf{X}) = \max_{\mathbf{X} \in \mathbf{D}_x} \min_{j \in [1:m]} Z_j(\mathbf{X}), \qquad (4.4)$$

где допустимая область $\mathbf{D}_x$ определяется только прямыми ограничениями на управляемые параметры x_i:

$$x_{i\,\min} < x_i < x_{i\,\max} .$$

Задачи оптимизации с учетом допусков. Содержательную сторону оптимизации с учетом допусков поясняет рис. 4.2, на котором представлены области работоспособности и допусковая в двумерном пространстве управляемых параметров. Если собственно допуски заданы и не относятся к управляемым параметрам, то цель оптимизации — максимальным образом совместить эти области так, чтобы вероятность выхода за пределы области работоспособности была минимальной.



Рис. 4.2. Области допусковая и работоспособности

Решение этой задачи исключительно трудоемко, так как на каждом шаге оптимизации нужно выполнять оценку упомянутой вероятности методами статистического анализа, а для сложных моделей объектов таким методом является метод статистических испытаний. Поэтому на практике подобные задачи решают, принимая те или иные допущения.

Например, если допустить, что цель оптимизации достигается при совмещении центров областей работоспособности **Э** и допусковой $\mathbf{X}_{\text{ном}}$, то оптимизация сводится к *задаче центрирования*, т.е. к определению центра **Э**. Задачу центрирования обычно решают путем предварительного нормирования управляемых параметров x_i с последующим вписыванием гиперкуба с максимально возможными размерами в нормированную область работоспособности.

Примечание. Нормирование проводят таким образом, что допусковая область приобретает форму гиперкуба, получающегося после нормирования.

Очевидно, что решение задачи центрирования позволяет не только оптимизировать номинальные значения проектных параметров, но и их допуски, если последние относятся к управляемым параметрам.

4.2. Обзор методов оптимизации

Классификация методов математического программирования. В САПР основными методами оптимизации являются поисковые методы. Поисковые методы основаны на пошаговом изменении управляемых параметров

$$\mathbf{X}_{k+1} = \mathbf{X}_k + \Delta\mathbf{X}_k, \qquad (4.5)$$

где в большинстве методов приращение $\Delta\mathbf{X}_k$ вектора управляемых параметров вычисляется по формуле

$$\Delta\mathbf{X}_k = h\mathbf{g}(\mathbf{X}_k). \qquad (4.6)$$

Здесь $\mathbf{X}_k$ — значение вектора управляемых параметров на k-м шаге, h — шаг, а $\mathbf{g}(\mathbf{X}_k)$ — направление поиска. Следовательно,. если выполняются условия сходимости, то реализуется пошаговое (итерационное) приближение к экстремуму.

Методы оптимизации классифицируют по ряду признаков.

В зависимости от числа управляемых параметров различают методы *одномерной* и *многомерной* оптимизации, в первых из них управляемый параметр единственный, во вторых размер вектора **X** не менее двух. Реальные задачи в САПР многомерны, методы одномерной оптимизации играют вспомогательную роль на отдельных этапах многомерного поиска.

Различают методы *условной* и *безусловной* оптимизации по наличию или отсутствию ограничений. Для реальных задач характерно наличие ограничений, однако методы безусловной оптимизации также представляют интерес, поскольку задачи условной оптимизации с помощью специальных методов могут быть сведены к задачам без ограничений.

В зависимости от числа экстремумов различают задачи одно- и многоэкстремальные. Если метод ориентирован на определение какого-либо локального экстремума, то такой метод относится к *локальным методам*. Если же результатом является глобальный экстремум, то метод называют *методом глобального поиска*. Удовлетворительные по вычислительной эффективности методы глобального поиска для общего случая отсутствуют и потому на практике в САПР используют методы поиска локальных экстремумов.

Наконец, в зависимости от того, используются при поиске производные целевой функции по управляемым параметрам или нет, различают методы нескольких порядков. Если производные не используются, то имеет место метод *нулевого порядка*, если используются первые или вторые производные, то соответственно метод *первого* или *второго порядка*. Методы первого порядка называют также градиентными, поскольку вектор первых производных $F(\mathbf{X})$ по **X** есть градиент целевой функции

$$\mathbf{grad}\,(F(\mathbf{X})) = (\partial F/\partial x_1,\ \partial F/\partial x_2, \ldots \partial F/\partial x_n).$$

Конкретные методы определяются следующими факторами:

1) способом вычисления направления поиска $\mathbf{g}(\mathbf{X}_k)$ в формуле (4.6);

2) способом выбора шага h;

3) способом определения окончания поиска.

Определяющим фактором является первый из перечисленных в этом списке, он подробно описан ниже.

Шаг может быть или постоянным, или выбираться исходя из одномерной оптимизации — поиска минимума целевой функции в выбранном направлении $\mathbf{g}(\mathbf{X}_k)$. В последнем случае шаг будем называть оптимальным.

Окончание поиска обычно осуществляют по правилу: если на протяжении r подряд идущих шагов траектория поиска остается в малой ε-окрестности текущей точки поиска $\mathbf{X}_k$, то поиск следует прекратить, следовательно, условие окончания поиска имеет вид

$$|\mathbf{X}_k - \mathbf{X}_{k-r}| < \varepsilon.$$

Методы одномерной оптимизации. К методам одномерной оптимизации относятся методы дихотомического деления, золотого сечения, чисел Фибоначчи, полиномиальной аппроксимации и ряд их модификаций.

Пусть задан отрезок $[A,B]$, на котором имеется один минимум (в общем случае нечетное число

минимумов). Согласно *методу дихотомического деления* (рис. 4.3,а) отрезок делят пополам и в точках, отстоящих от центра C отрезка на величину допустимой погрешности q, рассчитывают значения целевой функции $F(C+q)$ и $F(C-q)$. Если окажется, что $F(C+q) > F(C-q)$, то минимум находится на отрезке $[A,C]$, если $F(C+q) < F(C-q)$, то минимум — на $[C,B]$, если $F(C+q) = F(C-q)$ — на $[C-q,C+q]$. Таким образом, на следующем шаге вместо отрезка $[A,B]$ нужно исследовать суженный отрезок $[A,C]$, $[C,B]$ или $[C-q,C+q]$. Шаги повторяются, пока длина отрезка не уменьшится до величины погрешности q. Таким образом, требуется не более N шагов, где N — ближайшее к $\log((B-A)/q)$ целое значение, но на каждом шаге целевую функцию следует вычислять дважды.

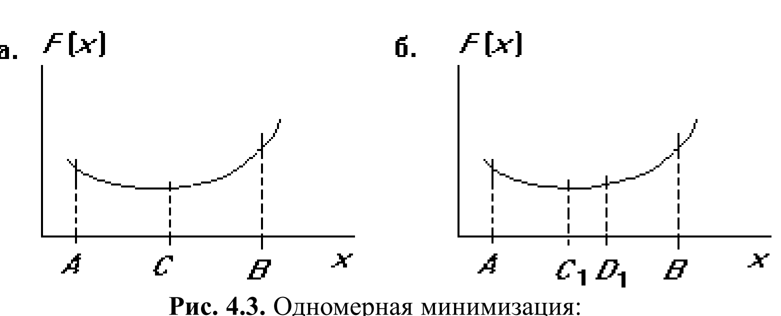


Рис. 4.3. Одномерная минимизация:
а - дихотомическое деление; б - золотое сечение

По *методу золотого сечения* (рис. 4.3,б) внутри отрезка $[A,B]$ выделяют две промежуточные точки C_1 и D_1 на расстоянии $s = aL$ от его конечных точек, где $L = B-A$ — длина отрезка. Затем вычисляют значения целевой функции $F(x)$ в точках C_1 и D_1. Если $F(C_1) < F(D_1)$, то минимум находится на отрезке $[A,D_1]$, если $F(C_1) > F(D_1)$), то — на отрезке $[C_1,B]$, если $F(C_1) = F(D_1)$ — на отрезке $[C_1, D_1]$. Следовательно, вместо отрезка $[A,B]$ теперь можно рассматривать отрезок $[A,D_1]$, $[C_1,B]$ или $[C_1, D_1]$, т.е. длина отрезка уменьшилась не менее чем в $L/(L-aL) = 1/(1-a)$ раз. Если подобрать значение a так, что в полученном отрезке меньшей длины одна из промежуточных точек совпадет с промежуточной точкой от предыдущего шага, т.е. в случае выбора отрезка $[A,D_1]$ точка D_2 совпадет с точкой C_1, а в случае выбора отрезка $[C_1,B]$ точка C_2 — с точкой D_1, то это позволит сократить число вычислений целевой функции на всех шагах (кроме первого) в 2 раза.

Условие получения такого значения a формулируется следующим образом $(1-2a)L_k = aL_{k-1}$, откуда с учетом того, что $L_k/L_{k-1} = 1/(1-a)$, имеем $a = 0{,}382$. Это значение и называют *золотым сечением*.

Таким образом, требуется не более N шагов и $N+1$ вычисление целевой функции, где N можно рассчитать, используя соотношение $(B-A)/E = (1-a)N$ при заданной погрешности E определения экстремума.

Согласно *методу чисел Фибоначчи,* используют числа Фибоначчи R_i, последовательность которых образуется по правилу $R_{i+2} = R_{i+1} + R_i$ при $R_0 = R_1 = 1$, т.е. ряд чисел Фибоначчи имеет вид 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144.... Метод аналогичен методу золотого сечения с тем отличием, что коэффициент a равен отношению R_{i-2}/R_i, начальное значение i определяется из условия, что R_i должно быть наименьшим числом Фибоначчи, превышающим величину $(B-A)/E$, где E — заданная допустимая погрешность определения экстремума. Так, если $(B-A)/E = 100$, то начальное значение $i = 12$, поскольку $R_1= 144$, и $a = 55/144 = 0{,}3819$, на следующем шаге будет $a = 34/89 = 0{,}3820$ и т.д.

По *методу полиномиальной аппроксимации* при аппроксимации $F(x)$ квадратичным полиномом

$$P(x) = a_0 + a_1x + a_2x^2 \qquad (4.7)$$

выбирают промежуточную точку C и в точках A, B, C вычисляют значения целевой функции. Далее решают систему из трех алгебраических уравнений, полученных подстановкой в (4.7) значений A,B,C вместо x и вычисленных значений функции вместо $P(x)$. В результате становятся известными значения коэффициентов a_k в (4.7) и, исходя из условия $dP(x)/dx = 0$, определяют экстремальную точку Э полинома. Например, если точка C выбрана в середине отрезка $[A,B]$, то $Э = C + (C-A)(F(A)-F(B)) / (2(F(A)-2F(C)+F(B)))$.

Методы безусловной оптимизации. Среди методов нулевого порядка в САПР находят применение методы Розенброка, конфигураций (Хука-Дживса), деформируемого многогранника (Нелдера-Мида), случайного поиска. К методам с использованием производных относятся методы наискорейшего спуска, сопряженных градиентов, переменной метрики.

Метод Розенброка является улучшенным вариантом покоординатного спуска.

Метод покоординатного спуска характеризуется выбором направлений поиска поочередно вдоль

всех n координатных осей, шаг рассчитывается на основе одномерной оптимизации, критерий окончания поиска $|\mathbf{X}_k - \mathbf{X}_{k\,n}| < \varepsilon$, где ε — заданная точность определения локального экстремума, n — размерность пространства управляемых параметров. Траектория покоординатного спуска для примера двумерного пространства управляемых параметров показана на рис. 4.4, где $\mathbf{X}_k$ — точки на траектории поиска, x_i — управляемые параметры. Целевая функция представлена своими линиями равного уровня, около каждой линии записано соответствующее ей значение $F(\mathbf{X})$. Очевидно, что **Э** есть точка минимума.

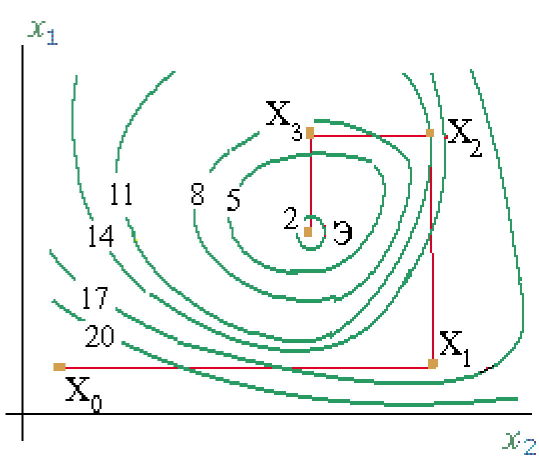


Рис. 4.4. Траектория покоординатного спуска

При использовании метода покоординатного спуска велика вероятность "застревания" поиска на дне оврага вдали от точки экстремума. На рис. 4.5 видно, что после попадания в точку **A**, расположенную на дне оврага, дальнейшие шаги возможны лишь в направлениях *aa* или *bb*, но они приводят к ухудшению целевой функции. Следовательно, поиск прекращается в точке **A**.

Примечание. *Оврагом* называют часть пространства управляемых параметров, в которой наблюдаются слабые изменения производных целевой функции по одним направлениям и значительные изменения с переменой знака — по некоторым другим направлениям. Знак производной меняется в точках, принадлежащих дну оврага.

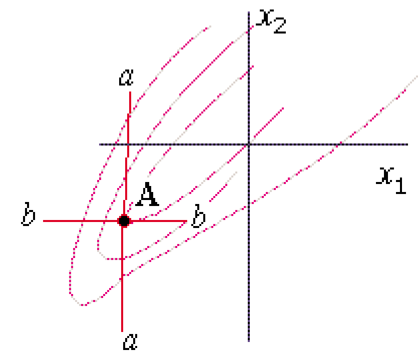


Рис. 4.5. "Застревание" покоординатного спуска на дне оврага

В то же время при благоприятной ориентации дна оврага, а именно при положении одной из координатных осей, близком к параллельности с дном оврага, поиск оказывается весьма быстрым. Эта ситуация показана на рис. 4.6.

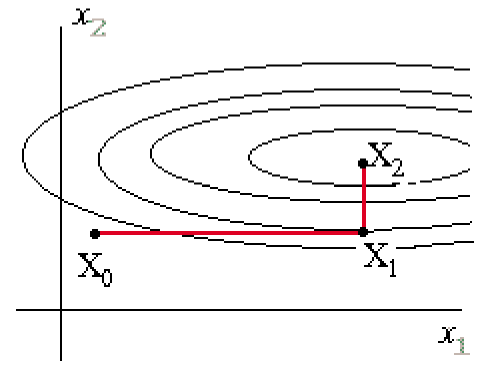


Рис. 4.6. Траектория покоординатного спуска при благоприятной ориентации координатных осей

Метод Розенброка заключается в таком повороте координатных осей, чтобы одна из них оказалась квазипараллельной дну оврага. Такой поворот осуществляют на основе данных, полученных после серии из n шагов покоординатного спуска. Положение новых осей s_i может быть получено линейным преобразованием прежних осей x_i: ось s_1 совпадает по направлению с вектором $\mathbf{X}_{k+n} - \mathbf{X}_k$; остальные оси выбирают из условия ортогональности к $\mathbf{X}_1$ и друг к другу.

Другой удачной модификацией покоординатного спуска является *метод конфигураций*. В соответствии с этим методом вначале выполняют обычную серию из n шагов покоординатного спуска, затем делают дополнительный шаг в направлении вектора $\mathbf{X}_k - \mathbf{X}_{k-n}$, как показано на рис. 4.7, где дополнительный шаг выполняют в направлении вектора $\mathbf{X}_3 - \mathbf{X}_1$, что и приводит в точку $\mathbf{X}_4$.

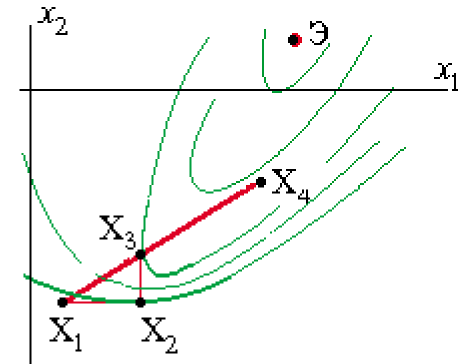


Рис. 4.7. Иллюстрация метода конфигураций

Поиск экстремума *методом деформируемого многогранника* основан на построении многогранника с $(n + 1)$ вершинами на каждом шаге поиска, где n — размерность пространства управляемых параметров. В начале поиска эти вершины выбирают произвольно, на последующих шагах выбор подчинен правилам метода.

Эти правила поясняются рис. 4.8 на примере двумерной задачи оптимизации. Выбраны вершины исходного треугольника: $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$. Новая вершина $\mathbf{X}_4$ находится на луче, проведенном из худшей вершины $\mathbf{X}_1$ (из вершины с наибольшим значением целевой функции) через центр тяжести **ЦТ** многогранника, причем рекомендуется $\mathbf{X}_4$ выбирать на расстоянии d от **ЦТ**, равном $|\mathbf{ЦТ}-\mathbf{X}_1|$. Новая вершина $\mathbf{X}_4$ заменяет худшую вер-

шину $\mathbf{X}_1$. Если оказывается, что $\mathbf{X}_4$ имеет лучшее значение целевой функции среди вершин многогранника, то расстояние d увеличивают. На рисунке именно эта ситуация имеет место и увеличение d дает точку $\mathbf{X}_5$. В новом многограннике с вершинами $\mathbf{X}_2$, $\mathbf{X}_3$, $\mathbf{X}_5$ худшей является вершина $\mathbf{X}_2$, аналогично получают вершину $\mathbf{X}_6$, затем вершину $\mathbf{X}_7$ и т.д. Если новая вершина окажется худшей, то в многограннике нужно сохранить лучшую вершину, а длины всех ребер уменьшить, например вдвое (стягивание многогранника к лучшей вершине). Поиск прекращается при выполнении условия уменьшения размеров многогранника до некоторого предела.

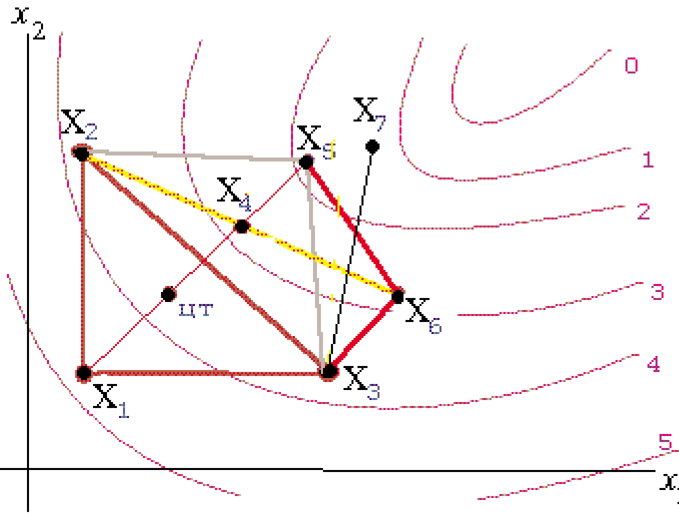


Рис. 4.8. Иллюстрация метода деформируемого многогранника

Случайные методы поиска характеризуются тем, что направления поиска **g** выбирают случайным образом.

Особенностью *метода наискорейшего спуска* является выполнение шагов поиска в градиентном направлении

$$\mathbf{X}_{k+1} = \mathbf{X}_k + h\,\mathbf{grad}\,F(\mathbf{X}) / |\mathbf{grad}\,F(\mathbf{X})|,$$

шаг h выбирается оптимальным с помощью одномерной оптимизации.

При использовании метода наискорейшего спуска, как и большинства других методов, эффективность поиска существенно снижается в овражных ситуациях. Траектория поиска приобретает зигзагообразный вид с медленным продвижением вдоль дна оврага в сторону экстремума. Чтобы повысить эффективность градиентных методов, используют несколько приемов.

Один из приемов, использованный в *методе сопряженных градиентов* (называемом также методом Флетчера-Ривса), основан на понятии сопряженности векторов. Векторы **A** и **B** называют **Q**-сопряженными, если $\mathbf{A}^{\mathrm{T}}\mathbf{Q}\mathbf{B} = 0$, где **Q** — положительно определенная квадратная матрица того же порядка, что и размер N векторов **A** и **B** (частный случай сопряженности — ортогональность векторов, когда **Q** является единичной матрицей порядка N), $\mathbf{A}^{\mathrm{T}}$ -вектор-строка, **B** — вектор-столбец.

Особенность сопряженных направлений для $\mathbf{Q} = \mathbf{\Gamma}$, где $\mathbf{\Gamma}$ — матрица Гессе, при в задачах с квадратичной целевой функцией $F(\mathbf{X})$ заключается в следующем: одномерная минимизация $F(\mathbf{X})$ последовательно по N сопряженным направлениям позволяет найти экстремальную точку не более, чем за N шагов.

Примечание. *Матрицей Гессе* называют матрицу вторых частных производных целевой функции по управляемым параметрам.

Основанием для использования поиска по $\mathbf{\Gamma}$-сопряженным направлениям является то, что для функций $F(\mathbf{X})$ общего вида может быть применена квадратичная аппроксимация, что на практике выливается в выполнение поиска более, чем за N шагов.

Пример. Поиск экстремума выполняют в соответствии с формулой

$$\mathbf{X}_i = \mathbf{X}_{i-1} + h\mathbf{S}_i. \qquad (4.8)$$

Направление $\mathbf{S}_{i+1}$ поиска на очередном шаге связано с направлением поиска $\mathbf{S}_i$ на предыдущем шаге соотношением

$$\mathbf{S}_{i+1} = -\,\mathbf{grad}F(\mathbf{X}_i) + w_i\mathbf{S}_i, \qquad (4.9)$$

где w_i — коэффициент. Кроме того, учитывают условие сопряженности

$$\mathbf{S}_{i+1}^{\mathrm{T}}\mathbf{\Gamma}\mathbf{S}_i = 0 \qquad (4.10)$$

и линейную аппроксимацию $\mathbf{grad}F(\mathbf{X})$ в окрестностях точки $\mathbf{X}_i$

$$\mathbf{grad}\,F(\mathbf{X}_{i+1}) = \mathbf{grad}\,F(\mathbf{X}_i) + \mathbf{\Gamma}(\mathbf{X}_{i+1} - \mathbf{X}_i). \qquad (4.11)$$

Поскольку шаг h рассчитывается исходя из условия одномерной оптимизации, то, во-первых, справедливо соотношение

$$\mathbf{S}_i^{\mathrm{T}}\mathbf{grad}\,F(\mathbf{X}_i) = 0, \qquad (4.12)$$

во-вторых, имеем

$$\mathbf{X}_i = \mathbf{X}_{i-1} + hw_{i-1}\,\mathbf{S}_{i-1} - h\,\mathbf{grad}\,F(\mathbf{X}_{i-1}),$$

откуда получаем

$$\partial F/\partial h = (\partial F(\mathbf{X})/\partial \mathbf{X})(\partial \mathbf{X}/\partial h) = \mathbf{grad}\,F(\mathbf{X}_i)\,\mathbf{grad}\,F(\mathbf{X}_{i-1}) = 0. \quad (4.13)$$

Алгоритм поиска сводится к применению формулы (4.9), пока не будет выполнено условие окончания вычислений

$$|\mathbf{grad}\,F(\mathbf{X}_k)| < \varepsilon.$$

Чтобы определить коэффициент w_i решают систему уравнений (4.8)-(4.13) путем подстановки в (4.10) величин $\mathbf{S}_{i+1}$ из (4.9) и $\mathbf{S}_i$ из (4.8)

$$\mathbf{S}_{i+1}^{\mathrm{T}}\mathbf{\Gamma}\mathbf{S}_i = (w_i\,\mathbf{S}_i - \mathbf{grad}\,F(\mathbf{X}_i))^{\mathrm{T}}\,\mathbf{\Gamma}(\mathbf{X}_i - \mathbf{X}_{i-1}) / h =$$
$$= (w_i\,\mathbf{S}_i - \mathbf{grad}\,F(\mathbf{X}_i))^{\mathrm{T}}\,\mathbf{\Gamma}\mathbf{\Gamma}^{-1}\,(\mathbf{grad}\,F(\mathbf{X}_i) - \mathbf{grad}\,F(\mathbf{X}_{i-1})) / h = 0;$$

или

$$(w_i\,\mathbf{S}_i - \mathbf{grad}\,F(\mathbf{X}_i))^{\mathrm{T}}\,(\mathbf{grad}\,F(\mathbf{X}_i) - \mathbf{grad}\,F(\mathbf{X}_{i-1})) = 0,$$

откуда

$$w_i\,\mathbf{S}_i^{\mathrm{T}}\,(\mathbf{grad}\,F(\mathbf{X}_i) - \mathbf{grad}\,F(\mathbf{X}_{i-1})) - \mathbf{grad}\,F(\mathbf{X}_i)^{\mathrm{T}}\,\mathbf{grad}\,F(\mathbf{X}_i) + \mathbf{grad}\,F(\mathbf{X}_i)^{\mathrm{T}}\,\mathbf{grad}\,F(\mathbf{X}_{i-1}) = 0$$

и с учетом (4.12) и (4.13)

$$w_i\,\mathbf{S}_i^{\mathrm{T}}\,\mathbf{grad}\,F(\mathbf{X}_{i-1}) + \mathbf{grad}\,F(\mathbf{X}_i)^{\mathrm{T}}\,\mathbf{grad}\,F(\mathbf{X}_i) = 0.$$

Следовательно,

$$w_i = \mathbf{grad}\,F(\mathbf{X}_i)^{\mathrm{T}}\,\mathbf{grad}\,F(\mathbf{X}_i)\,/\,\mathbf{S}_i^{\mathrm{T}}\,\mathbf{grad}\,F(\mathbf{X}_{i-1}) \quad (4.14)$$

На первом шаге поиска выбирают $\mathbf{S}_1 = —\,\mathbf{grad}\,F(\mathbf{X}_0)$ и находят точку $\mathbf{X}_1$. На втором шаге по формуле (4.14) рассчитывают w_1, по формулам (4.9) и (4.8) определяют $\mathbf{S}_2$ и $\mathbf{X}_2$ и т.д.

Метод переменной метрики (иначе метод Девидона-Флетчера-Пауэлла) можно рассматривать как результат усовершенствования метода второго порядка — метода Ньютона.

Метод Ньютона основан на использовании необходимых условий безусловного экстремума целевой функции $F(\mathbf{X})$

$$\mathbf{grad}\,F(\mathbf{X}) = 0. \quad (4.15)$$

Выражение (4.15) представляет собой систему алгебраических уравнений, для решения которой можно применить известный численный метод, называемый методом Ньютона. Корень системы (4.15) есть стационарная точка, т.е. возможное решение экстремальной задачи. Метод Ньютона является итерационным, он основан на линеаризации (4.15) в окрестности текущей точки поиска $\mathbf{X}_k$

$$\mathbf{grad}\,F(\mathbf{X}) = \mathbf{grad}\,F(\mathbf{X}_k) + \mathbf{\Gamma}(\mathbf{X}-\mathbf{X}_k) = 0. \quad (4.16)$$

Выражение (4.16) — это система линейных алгебраических уравнений. Ее корень есть очередное приближение $\mathbf{X}_{k+1}$ к решению $\mathbf{X}_{k+1} = \mathbf{X}_k - \mathbf{\Gamma}^{-1}(\mathbf{X}_k)\,\mathbf{grad}\,F(\mathbf{X}_k)$.

Если процесс сходится, то решение достигается за малое число итераций, окончанием которых служит выполнение условия

$$|\mathbf{X}_{k+1} - \mathbf{X}_k| < \varepsilon.$$

Главный недостаток метода — высокая трудоемкость вычисления и обращения матрицы $\mathbf{\Gamma}$, к тому же ее вычисление численным дифференцированием сопровождается заметными погрешностями, что снижает скорость сходимости.

В методе переменной метрики вместо трудно вычисляемой обратной матрицы Гессе используют некоторую более легко вычисляемую матрицу $\mathbf{N}$, т.е.

$$\mathbf{X}_{k+1} = \mathbf{X}_k + \mathbf{N}\,\mathbf{grad}\,F(\mathbf{X}_k).$$

Введем обозначения:

$$d\mathbf{g}_k = \mathbf{grad}\,F(\mathbf{X}_k) - \mathbf{grad}\,F(\mathbf{X}_{k-1});$$
$$d\mathbf{X}_k = \mathbf{X}_k - \mathbf{X}_{k-1};$$

$\mathbf{E}$ — единичная матрица. Начальное значение матрицы $\mathbf{N}_0 = \mathbf{E}$. Матрицу $\mathbf{N}$ корректируют на каждом шаге, т.е.

$$\mathbf{N}_{k+1} = \mathbf{N}_k + \mathbf{A}_k{}^{\mathrm{T}}\mathbf{B}_k,$$

где

$$\mathbf{A}_k = d\mathbf{X}_k\, d\mathbf{X}_k{}^{\mathrm{T}} / (d\mathbf{X}^{\mathrm{T}} d\mathbf{g}_k),$$
$$\mathbf{B}_k = \mathbf{N}_k\, d\mathbf{g}_k\, d\mathbf{g}_k{}^{\mathrm{T}}\, \mathbf{N}_k{}^{\mathrm{T}} / (d\mathbf{g}_k{}^{\mathrm{T}}\mathbf{N}_k\, d\mathbf{g}_k).$$

Поэтому

$$\mathbf{N}_{k+1} = \mathbf{E} + \sum_{i=0}^{k} \mathbf{A}_i - \sum_{i=0}^{k} \mathbf{B}_i.$$

Можно показать, что $\mathbf{A}_i$ стремится к $\mathbf{Г}^{-1}$, $\mathbf{B}_i$ _ к $\mathbf{E}$ при $k \to n$, где n — размерность пространства управляемых параметров. Спустя n шагов, нужно снова начинать с $\mathbf{N}_{n+1} = \mathbf{E}$.

Необходимые условия экстремума. В задачах безусловной оптимизации необходимые условия представляют собой равенство нулю градиента целевой функции

$$\mathbf{grad}\, F(\mathbf{X}) = 0.$$

В общей задаче математического программирования (4.1) необходимые условия экстремума, называемые условиями Куна-Таккера, формулируются следующим образом:

Для того чтобы точка **Э** была экстремальной точкой выпуклой задачи математического программирования (ЗМП), необходимо наличие неотрицательных коэффициентов u_i, таких, что

$$u_i\, \varphi_i(\mathbf{Э}) = 0, \quad i = 1,2,...m; \tag{4.17}$$

и при этом соблюдались ограничения задачи, а также выполнялось условие

$$\mathbf{grad}\, \mathrm{F}(\mathbf{Э}) + \sum_{i=1}^{m} \mathrm{u_i}\, \mathbf{grad}\, \varphi_i(\mathbf{Э}) + \sum_{j=1}^{L} \mathrm{a_j}\, \psi_j(\mathbf{Э}) = 0, \tag{4.18}$$

где m — число ограничений типа неравенств, L — то же равенств, коэффициенты $a_j > 0$.

За приведенной абстрактной формулировкой условий скрывается достаточно просто понимаемый геометрический смысл. Действительно, рассмотрим сначала случай с ограничениями только типа неравенств. Если максимум находится внутри допустимой области **R**, то, выбирая все $u_i = 0$, добиваемся выполнения (4.17); если же точка максимума **Э** лежит на границе области **R**, то, как видно из левой части рис. 4.9, эту точку всегда соответствующим подбором неотрицательных u_i можно поместить внутрь оболочки, натянутой на градиенты целевой функции $F(\mathbf{X})$ и функций-ограничений $\varphi_i(\mathbf{X})$. Наоборот, если точка не является экстремальной, то (4.17) нельзя выполнить при любом выборе положительных коэффициентов u_i (см. правую часть рис. 4.9, где рассматриваемая точка **X** лежит вне выпуклой оболочки, натянутой на градиенты). Учет ограничений типа равенств очевиден, если добавляется последняя из указанных в (4.18) сумма.

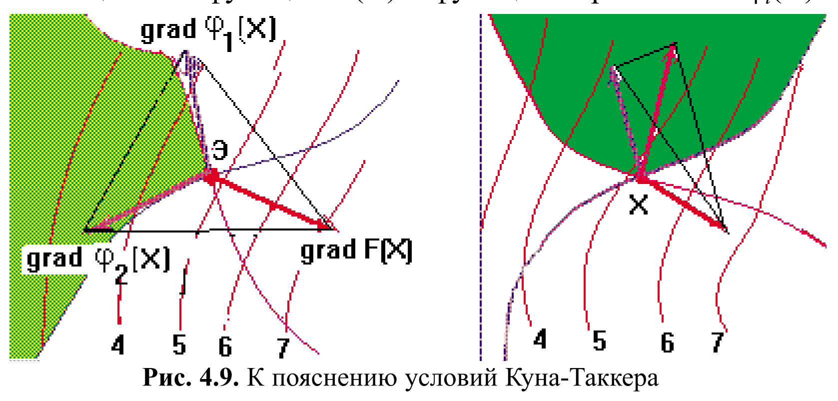


Рис. 4.9. К пояснению условий Куна-Таккера

Методы поиска условных экстремумов. Широко известен метод множителей Лагранжа, ориентированный на поиск экстремума при наличии ограничений типа равенств $\boldsymbol{\psi}(\mathbf{X}) = 0$, т.е. на решение задачи

$$\underset{\mathbf{X} \in \mathbf{R}}{\mathbf{extr}}\, F(\mathbf{X}), \tag{4.19}$$

где $\mathbf{R} = \{ \mathbf{X} \mid \boldsymbol{\psi}(\mathbf{X}) = 0 \}$.

Суть метода заключается в преобразовании задачи условной оптимизации (4.19) в задачу безусловной оптимизации с помощью образования новой целевой функции

$$\Phi(\mathbf{X},\mathbf{V}) = \mathrm{F}(\mathbf{X}) + \sum_{i=1}^{L} \lambda_i\, \psi_i(\mathbf{X}),$$

где $\mathbf{\Lambda} = (\lambda_1, \lambda_2, \lambda_3 ... \lambda_h)$ — вектор множителей Лагранжа, L — число ограничений.

Необходимые условия экстремума функции Ф(**X**):

$$\partial\Phi(\mathbf{X},\mathbf{\Lambda})/\partial\mathbf{X} = \partial F(\mathbf{X})/\partial\mathbf{X} + \sum_{i=1}^{L} \lambda_i \, \partial\psi_i(\mathbf{X})/\partial\mathbf{X} = 0;$$
$$\partial\Phi(\mathbf{X},\mathbf{\Lambda})/\partial\mathbf{\Lambda} = \boldsymbol{\psi}(\mathbf{X}) = 0. \qquad (4.20)$$

Система (4.20) содержит $n+L$ алгебраических уравнений, где n — размерность пространства управляемых параметров, ее решение дает искомые координаты экстремальной точки и значения множителей Лагранжа. Однако при численном решении (4.20), что имеет место при использовании алгоритмических моделей, возникают те же трудности, что и в методе Ньютона. Поэтому в САПР основными методами решения ЗМП являются методы штрафных функций и проекции градиента.

Основная идея *методов штрафных функций* — преобразование задачи условной оптимизации в задачу безусловной оптимизации путем формирования новой целевой функции Ф(**X**) введением в исходную целевую функцию F(**X**) специальным образом выбранной функции штрафа S(**X**):

$$\Phi(\mathbf{X}) = F(\mathbf{X}) + rS(\mathbf{X}),$$

где r — множитель, значения которого можно изменять в процессе оптимизации.

Среди методов штрафных функций различают методы внутренней и внешней точки. Согласно методам внутренней точки (иначе называемым методами *барьерных функций*) исходную для поиска точку можно выбирать только внутри допустимой области, а для методов внешней точки как внутри, так и вне допустимой области (важно лишь, чтобы в ней функции целевая и ограничений были бы определены). Ситуация появления барьера у целевой функции Ф(x) и соотношение между условным в точке x_2 и безусловным в точке x_1 минимумами F(x) в простейшем одномерном случае иллюстрируется рис. 4.10.

Рис. 4.10. Пояснение метода штрафных функций

Примеры штрафных функций:

1) для метода внутренней точки при ограничениях $\varphi_i(\mathbf{X}) > 0$

$$S(\mathbf{X}) = \sum_{i=1}^{m} (1/\varphi_i(\mathbf{X})),$$

где m — число ограничений типа неравенств;

2) для метода внешней точки при таких же ограничениях

$$S(\mathbf{X}) = \sum_{i=1}^{m} (\min\{0, \varphi_i(\mathbf{X})\})^2 \text{ —}$$

здесь штраф сводится к включению в Ф(**X**) суммы квадратов активных (т.е. нарушенных) ограничений;

3) в случае ограничений типа равенств $\psi_i(\mathbf{X}) = 0$

$$S(\mathbf{X}) = \sum_{i=1}^{L} (\psi_i(\mathbf{X}))^2.$$

Чем больше коэффициент r, тем точнее решение задачи, однако при больших r может ухудшаться ее обусловленность. Поэтому в начале поиска обычно выбирают умеренные значения r, увеличивая их в окрестностях экстремума.

Основной вариант *метода проекции градиента* ориентирован на задачи математического программирования с ограничениями типа равенств.

Поиск при выполнении ограничений осуществляется в подпространстве (n-m) измерений, где n — число управляемых параметров, m — число ограничений, при этом движение осуществляется в направлении проекции градиента целевой функции F(**X**) на гиперплоскость, касательную к гиперповерхности ограничений (точнее к гиперповерхности пересечения гиперповерхностей ограничений).

Поиск минимума начинают со спуска из исходной точки на гиперповерхность ограничений. Далее выполняют шаг в указанном выше направлении (шаг вдоль гиперповерхности ограничений). Поскольку этот шаг может привести к заметному нарушению ограничений, вновь повторяют спуск на ги-

перповерхность ограничений и т.д. Другими словами, поиск заключается в выполнении пар шагов, каждая пара включает спуск на гиперповерхность ограничений и движение вдоль гиперповерхности ограничений.

Идею метода легко пояснить для случая поиска в двумерном пространстве при одном ограничении ψ(**X**) = 0. На рис. 4.11 это ограничение представлено жирной линией, а целевая функция — совокупностью более тонких линий равного уровня. Спуск обычно осуществляют по нормали к гиперповерхности ограничений (в данном случае к линии ограничения). Условие окончания поиска основано на сопоставлении значений целевой функции в двух последовательных точках, получаемых после спуска на гиперповерхность ограничений.

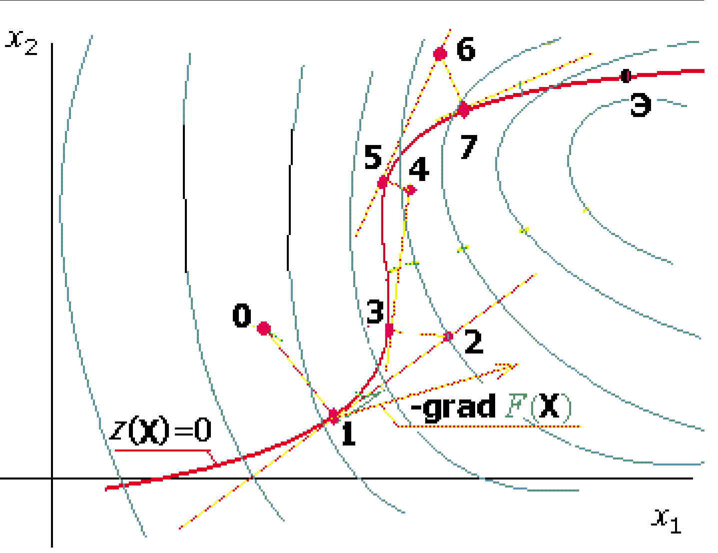


Рис. 4.11. Траектория поиска в соответствии с методом проекции градиента: Э - условный экстремум; **0, 1, 2, 3** - точки на траектории поиска

Рассмотрим вопрос, касающийся получения аналитических выражений для направлений спуска и движения вдоль гиперповерхности ограничений.

Спуск. Необходимо из текущей точки поиска **B** попасть в точку **A**, являющуюся ближайшей к **B** точкой на гиперповерхности ограничений, т.е. решить задачу

$$\min |\mathbf{B}-\mathbf{A}|$$

при условии ψ(**X**)=0, которое после линеаризации в окрестностях точки **B** имеет вид

$$\psi(\mathbf{B}) + (\mathbf{grad}\ \psi(\mathbf{B}))^T(\mathbf{A}-\mathbf{B}) = 0.$$

Используя метод множителей Лагранжа, обозначая **A-B=U** и учитывая, что минимизация расстояния равнозначна минимизации скалярного произведения **U** на **U**, запишем

$$\Phi(\mathbf{A}) = \mathbf{U}^T\mathbf{U} + \lambda\ (\psi(\mathbf{B})+(\mathbf{grad}\ \psi(\mathbf{B}))^T\mathbf{U});$$

$$\partial\Phi/\partial\mathbf{A} = 2\mathbf{U} + \lambda\ (\mathbf{grad}\ \psi(\mathbf{B})) = 0; \qquad (4.21)$$

$$\partial\Phi/\partial\lambda = \psi(\mathbf{B}) + (\mathbf{grad}\ \psi(\mathbf{B}))^T\mathbf{U} = 0. \qquad (4.22)$$

Тогда из (4.21) получаем выражение

$$\mathbf{U} = -\ 0{,}5\lambda\ (\mathbf{grad}\ \psi(\mathbf{B})),$$

подставляя его в (4.22), имеем

$$\psi(\mathbf{B}) - 0{,}5\lambda\ (\mathbf{grad}\ \psi(\mathbf{B}))^T\ \mathbf{grad}\ \psi(\mathbf{B}) = 0;$$

откуда

$$\lambda = (0{,}5(\mathbf{grad}\ \psi(\mathbf{B}))^T\ \mathbf{grad}\ \psi(\mathbf{B}))^{-1}\psi(\mathbf{B}).$$

и окончательно, подставляя λ в (4.21), находим

$$\mathbf{U} = -\ \mathbf{grad}\ \psi(\mathbf{B})(\mathbf{grad}\ \psi(\mathbf{B}))^T\ \mathbf{grad}\ \psi(\mathbf{B}))^{-1}\ \psi(\mathbf{B}).$$

Движение вдоль гиперповерхности ограничений. Шаг в гиперплоскости **D**, касательной к гиперповерхности ограничений, следует сделать в направлении вектора **S**, на котором целевая функция уменьшается в наибольшей мере при заданном шаге *h*. Уменьшение целевой функции при переходе из точки **A** в новую точку **C** подсчитывают, используя формулу линеаризации *F*(**X**) в окрестностях точки **A:**

$$F(\mathbf{C}) - F(\mathbf{A}) = h(\mathbf{grad}\ F(\mathbf{A}))^T\mathbf{S},$$

где **grad** *F*(**A**)^T**S** — приращение *F*(**X**), которое нужно минимизировать, варьируя направления **S**

$$\mathbf{min}\ F(\mathbf{C}) = \mathbf{min}\ ((\mathbf{grad}\ F(\mathbf{A}))^T\mathbf{S}), \qquad (4.23)$$

где вариация **S** осуществляется в пределах гиперплоскости **D**; **grad**ψ(**A**) и **S** — ортогональные векто-

ры. Следовательно, минимизацию (4.23) необходимо выполнять при ограничениях

$$(\mathbf{grad}\ \psi(\mathbf{A}))^{\mathrm{T}}\mathbf{S} = 0,$$
$$\mathbf{S}^{\mathrm{T}}\mathbf{S} = 1.$$

Последнее ограничение говорит о том, что при поиске направления движения, вектор **S** должен лишь указывать это направление, т.е. его длина несущественна (пусть **S** — единичный вектор).

Для решения (4.23) используем метод множителей Лагранжа

$$\Phi(\mathbf{S},\lambda,q) = (\mathbf{grad}\ F(\mathbf{A}))^{\mathrm{T}}\mathbf{S} + \lambda(\mathbf{grad}\ \psi(\mathbf{A}))^{\mathrm{T}}\mathbf{S} + q(\mathbf{S}^{\mathrm{T}}\mathbf{S}\text{-}1),$$

где λ и q — множители Лагранжа;

$$\partial\Phi/\partial\mathbf{S} = \mathbf{grad}\ F(\mathbf{A}) + \lambda\ \mathbf{grad}\ \psi(\mathbf{A}) + q\mathbf{S} = 0; \quad (4.24)$$

$$\partial\Phi/\partial\lambda = (\mathbf{grad}\ \psi(\mathbf{A}))^{\mathrm{T}}\mathbf{S} = 0; \quad (4.25)$$

$$\partial\Phi/\partial q = \mathbf{S}^{\mathrm{T}}\mathbf{S}\text{-}1 = 0. \quad (4.26)$$

Из (4.24) следует, что

$$\mathbf{S} = -(\mathbf{grad}\ F(\mathbf{A}) + \lambda\ \mathbf{grad}\ \psi(\mathbf{A})\ )/q;$$

подставляя **S** в (4.25), получаем

$$(\mathbf{grad}\ \psi(\mathbf{A}))^{\mathrm{T}}\ \mathbf{grad}\ F(\mathbf{A}) + \lambda\ (\mathbf{grad}\ \psi(\mathbf{A}))^{\mathrm{T}}\ \mathbf{grad}\ \psi(\mathbf{A}) = 0,$$

откуда

$$\lambda = -\ [(\mathbf{grad}\ \psi(\mathbf{A}))^{\mathrm{T}}\ \mathbf{grad}\ \psi(\mathbf{A})]^{-1}\ (\mathbf{grad}\ \psi(\mathbf{A}))^{\mathrm{T}}\ \mathbf{grad}\ F(\mathbf{A}),\ \mathbf{S} =$$
$$= -\ \{\mathbf{grad}F(\mathbf{A})\text{-}\mathbf{grad}\psi(\mathbf{A})[(\mathbf{grad}\psi(\mathbf{A}))^{\mathrm{T}}\mathbf{grad}\psi(\mathbf{A})]^{-1}(\mathbf{grad}\psi(\mathbf{A}))^{\mathrm{T}}\mathbf{grad}F(\mathbf{A})\}\ /\ q =$$
$$= -\ \{\mathbf{E}\ \text{-}\ \mathbf{grad}\ \psi(\mathbf{A})[(\mathbf{grad}\ \psi(\mathbf{A}))^{\mathrm{T}}\ \mathbf{grad}\ \psi(\mathbf{A})]^{-1}\ (\mathbf{grad}\ \psi(\mathbf{A}))^{\mathrm{T}}\}\mathbf{grad}\ F(\mathbf{A})\ /\ q. \quad (4.27)$$

Таким образом, матрица

$$\mathbf{P} = \mathbf{E}\ \text{-}\ \mathbf{grad}\ \psi(\mathbf{A})[(\mathbf{grad}\ \psi(\mathbf{A}))^{\mathrm{T}}\ \mathbf{grad}\ \psi(\mathbf{A})]^{-1}\ \mathbf{grad}\ \psi(\mathbf{A}))^{\mathrm{T}}$$

представляет собой проектирующую матрицу, а вектор **S**, рассчитанный по (4.27), — проекцию градиента **grad**F(**A**) на гиперповерхность ограничений.

Частным случаем применения метода проекции градиента являются задачи оптимизации с максиминным критерием. Действительно, для поиска экстремума функции минимума

$$\max_{\mathbf{X}}\ \min_{j}\ Z_j(\mathbf{X}),$$

где Z_j — нормированная величина j-го выходного параметра y_j, удобно применять метод проекции градиента. В качестве ограничений задачи в исходной постановке фигурируют только прямые ограничения

$$x_{\max\,i} > x_i > x_{\min\,i}.$$

Здесь $x_{\max i}$ и $x_{\min i}$— граничные значения допустимого диапазона варьирования параметра x_i. В процессе поиска, если минимальной является функция Z_q(**X**) и траектория поиска пересекает гребень

$$Z_q(\mathbf{X})\ \text{-}\ Z_k(\mathbf{X}) = 0, \quad (4.28)$$

то поиск продолжается в направлении проекции градиента функции Z_q(**X**) на гиперповерхность гребня (4.28).

4.3. Постановка задач структурного синтеза

Процедуры синтеза проектных решений. Принятие проектных решений охватывает широкий круг задач и процедур — от выбора вариантов в конечных и обозримых множествах до задач творческого характера, не имеющих формальных способов решения.

Соответственно в САПР применяют как средства формального синтеза проектных решений, выполняемого в автоматическом режиме, так и вспомогательные средства, способствующие выполнению синтеза проектных решений в интерактивном режиме. К вспомогательным средствам относятся базы типовых проектных решений, системы обучения проектированию, программно-методические комплексы верификации проектных решений, унифицированные языки описания ТЗ и результатов проектирования.

Задачи синтеза структур проектируемых объектов относятся к наиболее трудно формализуемым. Существует ряд общих подходов к постановке этих задач, однако практическая реализация большинства из них неочевидна. Поэтому имеются лишь “островки” автоматического выполнения процедур синтеза среди “моря” проблем, ждущих автоматизации.

Именно по этой причине структурный синтез, как правило, выполняют в интерактивном режиме при решающей роли инженера-разработчика, а ЭВМ играет вспомогательную роль: предоставление необходимых справочных данных, фиксация и оценка промежуточных и окончательных результатов.

Однако в ряде приложений имеются и примеры успешной автоматизации структурного синтеза в ряде приложений; среди них заслуживают упоминания в первую очередь задачи конструкторского проектирования печатных плат и кристаллов БИС, логического синтеза комбинационных схем цифровой автоматики и вычислительной техники, синтеза технологических процессов и управляющих программ для механообработки в машиностроении и некоторые другие.

Структурный синтез заключается в преобразовании описаний проектируемого объекта: исходное описание содержит информацию о требованиях к свойствам объекта, об условиях его функционирования, ограничениях на элементный состав и т.п., а результирующее описание должно содержать сведения о *структуре*, т.е. о составе элементов и способах их соединения и взаимодействия.

Постановки и методы решения задач структурного синтеза в связи с трудностями формализации не достигли степени обобщения и детализации, свойственной математическому обеспечению процедур анализа. Достигнутая степень обобщения выражается в установлении типичной последовательности действий и используемых видов описаний при их преобразованиях в САПР. Исходное описание, как правило, представляет собой ТЗ на проектирование, по нему составляют описание на некотором формальном языке, являющемся входным языком используемых подсистем САПР. Затем выполняют преобразования описаний, и получаемое итоговое для данного этапа описание документируют — представляют в виде твердой копии или файла в соответствующем формате для передачи на следующий этап.

Важное значение для развития подсистем синтеза в САПР имеют разработка и унификация языков представления описаний (спецификаций). Каждый язык, поддерживая выбранную методику принятия решений, формирует у пользователей САПР — разработчиков технических объектов определенный стиль мышления; особенности языков непосредственно влияют на особенности правил преобразования спецификаций. Примерами унифицированных языков описания проектных решений являются язык VHDL для радиоэлектроники, он сочетает в себе средства для функциональных, поведенческих и структурных описаний, или язык Express — универсальный язык спецификаций для представления и обмена информацией в компьютерных средах.

Задача принятия решений. Имеется ряд подходов для обобщенного описания задач принятия проектных решений в процессе структурного синтеза.

Задачу принятия решений (ЗПР) формулируют следующим образом:

ЗПР = <**А, К**, Мод,П>,

где **А** — множество альтернатив проектного решения, **К** = $(K_1, K_2,...,K_m)$ — множество критериев (выходных параметров), по которым оценивается соответствие альтернативы поставленным целям; Мод: **А→К** — модель, позволяющая для каждой альтернативы рассчитать вектор критериев, П — решающее правило для выбора наиболее подходящей альтернативы в многокритериальной ситуации.

В свою очередь, каждой альтернативе конкретного приложения можно поставить в соответствие значения упорядоченного множества (набора) атрибутов **X** = $<x_1, x_2,...x_n>$, характеризующих свойства альтернативы. При этом x_i может быть величиной типа real, integer, boolean, string (в последнем случае величину называют *предметной* или *лингвистической*). Множество **X** называют *записью* (в теории баз данных), *фреймом* (в искусственном интеллекте) или *хромосомой* (в генетических алгоритмах). Модель Мод называют структурно-критериальной, если среди x_i имеются параметры, характеризующие структуру моделируемого объекта.

Основными проблемами в ЗПР являются:

— компактное представление множества вариантов (альтернатив);

— построение модели синтезируемого устройства, в том числе выбор степени абстрагирования для оценки значений критериев;

— формулировка предпочтений в многокритериальных ситуациях (т.е. преобразование векторного критерия **К** в скалярную целевую функцию);

— установление порядка (предпочтений) между альтернативами в отсутствие количественной оценки целевой функции (что обычно является следствием неколичественного характера всех или части критериев);

— выбор метода поиска оптимального варианта (сокращение перебора вариантов).

Присущая проектным задачам неопределенность и нечеткость исходных данных, а иногда и моделей, диктуют использование специальных методов количественной формулировки исходных неколичественных данных и отношений. Эти специальные методы либо относятся к области построения измерительных шкал, либо являются предметом теории нечетких множеств.

Измерительные шкалы могут быть:

1) абсолютными:

2) номинальными (классификационными), значения шкалы представляют классы эквивалентности, примером может служить шкала цветов; такие шкалы соответствуют величинам неколичественного характера;

3) порядковыми, если между объектами *A* и *B* установлено одно из следующих отношений: простого порядка, гласящее, что если *A* лучше *B*, то *B* хуже *A*, и соблюдается транзитивность; или слабого порядка, т.е. либо *A* не хуже *B*, либо *A* не лучше *B*; или частичного порядка. Для формирования целевой функции $F(\mathbf{X})$ производится оцифровка порядковой шкалы, т.е. при минимизации, если *A* предпочтительнее *B*, то $F(\mathbf{X}_a)<F(\mathbf{X}_b)$, где $\mathbf{X}_a$ и $\mathbf{X}_b$ — множества атрибутов объектов *A* и *B* соответственно;

4) интервальными, отражающими количественные отношения интервалов: шкала единственна с точностью до линейных преобразований, т.е. $y=ax+b$, $a>0$, $-\infty<b<\infty$, или $y=ax$ при $a\neq 0$, или $y=x+b$.

В большинстве случаев структурного синтеза математическая модель в виде алгоритма, позволяющего по заданному множеству **X** и заданной структуре объекта рассчитать вектор критериев **К**, оказывается известной. Например, такие модели получаются автоматически в программах анализа типа *Spice*, *Adams* или ПА-9 для объектов, исследуемых на макроуровне. Однако в ряде других случаев такие модели неизвестны в силу недостаточной изученности процессов и их взаимосвязей в исследуемой среде, но известна совокупность результатов наблюдений или экспериментальных исследований. Тогда для получения моделей используют специальные *методы идентификации и аппроксимации* (модели, полученные подобным путем иногда называют феноменологическими).

Среди методов формирования моделей по экспериментальным данным наиболее известны *методы планирования экспериментов*. Не менее популярным становится подход, основанный на использовании *искусственных нейронных сетей*.

Если же математическая модель $\mathbf{X} \to \mathbf{K}$ остается неизвестной, то стараются использовать подход на базе *систем искусственного интеллекта* (*экспертных систем*).

Возможности практического решения задач *дискретного математического программирования* (ДМП) изучаются в теории сложности задач выбора, где показано, что задачи даже умеренного размера, относящиеся к классу **NP**-полных задач, в общем случае удается решать только приближенно.

Поэтому большинство практических задач структурного синтеза решают с помощью приближенных (эвристических) методов. Это методы, использующие специфические особенности того или иного класса задач и не гарантирующие получения оптимального решения. Часто они приводят к результатам, близким к оптимальным, при приемлемых затратах вычислительных ресурсов.

Если все управляемые параметры альтернатив, обозначаемые в виде множества **X**, являются количественными оценками, то используют *приближенные методы* оптимизации. Если в **X** входят также параметры неколичественного характера и пространство **X** неметризуемо, то перспективными являются *эволюционные методы* вычислений, среди которых наиболее развиты *генетические методы*. Наконец, в отсутствие обоснованных моделей Мод их создают, основываясь на экспертных знаниях в виде некоторой системы искусственного интеллекта.

Представление множества альтернатив. Решению проблем упорядочения и описания множества альтернатив и связей между ними в конкретных приложениях посвящена специальная область знания, которую по аналогии с наукой описания множеств животных и растений в биологии можно назвать *систематикой*.

Простейший способ задания множества **A** — явное перечисление всех альтернатив. Семантика и форма описания альтернатив существенно зависят от приложения. Для представления таких описаний в памяти ЭВМ и доступа к ним используют *информационно-поисковые системы* (ИПС). Каждой альтернативе в ИПС соответствует поисковый образ, состоящий из значений атрибутов x_i и ключевых слов вербальных характеристик.

Явное перечисление альтернатив при представлении множества альтернатив возможно лишь при малой мощности **A**. Поэтому в большинстве случаев используют неявное описание **A** в виде способа (алгоритма или набора правил **P**) синтеза проектных решений из ограниченного набора элементов **Э**. Поэтому здесь **A** = <**P**,**Э**>, а типичный процесс синтеза проектных решений состоит из следующих этапов:

1) формирование альтернативы A_i (это может быть выбор из базы данных ИПС по сформированному поисковому предписанию или генерация из **Э** в соответствии с правилами **P**);

2) оценка альтернативы по результатам моделирования с помощью модели Мод;

3) принятие решения (выполняется ЛПР — лицом, принимающим решение, или автоматически) относительно перехода к следующей альтернативе или прекращения поиска.

Для описания множеств **P** и **Э** используют следующие подходы.

1. *Морфологические таблицы и альтернативные И-ИЛИ-деревья*.

2. Представление знаний в *интеллектуальных системах* — фреймы, семантические сети, продукции.

3. *Генетические методы*.

4. Базы *физических эффектов* и *эвристических приемов*, применяемые при решении задач изобретательского характера.

Морфологические таблицы. *Морфологическая таблица* (**M**) представляет собой обобщенную структуру в виде множества функций, выполняемых компонентами синтезируемых объектов рассматриваемого класса, и подмножеств способов их реализации. Каждой функции можно поставить в соответствие одну строку таблицы, каждому способу ее реализации — одну клетку в этой строке. Следовательно, в морфологических таблицах элемент M_{ij} означает j-й вариант реализации i-й функции в классе технических объектов, описываемом матрицей **M**.

Другими словами, множество альтернатив можно представить в виде отношения **M,** называемого морфологической таблицей

$$\mathbf{M} = \langle \mathbf{X}, \mathbf{R} \rangle,$$

где **X** — множество свойств (характеристик или функций), присущих объектам рассматриваемого типа, n — число этих свойств, $\mathbf{R} = \langle \mathbf{R}_1, \mathbf{R}_2, ..., \mathbf{R}_n \rangle$, $\mathbf{R}_i$ — множество значений (способов реализации) i-го свойства, мощность этого множества далее обозначена N_i. При этом собственно множество альтернатив **A** представлено композицией множеств $\mathbf{R}_i$, т.е. каждая альтернатива включает по одному элементу (значению) из каждой строки морфологической таблицы. Очевидно, что общее число альтернатив k, представляемых морфологической таблицей, равно

$$k = \prod_{i=1}^{n} N_i.$$

Морфологические таблицы обычно считают средством неавтоматизированного синтеза, помогающим человеку просматривать компактно представленные альтернативы, преодолевать психологическую инерцию. Последнее связано с тем, что внимание ЛПР обращается на варианты, которые без морфологической таблицы оставались бы вне его поля зрения.

Собственно таблица **M** не содержит сведений о способе синтеза. Однако на базе **M** возможно построение методов синтеза с элементами алгоритмизации. В таких методах вводится метризация мор-

фологического пространства. Морфологическое пространство составляют возможные законченные структуры, принимается, что расстояние между структурами $\mathbf{C}_1$ и $\mathbf{C}_2$ есть число несовпадающих элементов (каждая клетка **М** есть один элемент). Поэтому можно говорить об окрестностях решений. Далее исходят из предположения о компактности "хороших" решений, которое позволяет вместо полного перебора ограничиваться перебором в малой окрестности текущей точки поиска. Таким образом, гипотеза о "компактности" и метризация пространства решений фактически приводят к построению математической модели, к которой можно применить методы дискретной оптимизации, например локальные методы.

К недостаткам **М** относятся неучет запрещенных сочетаний элементов в законченных структурах и отражение состава элементов в структурах без конкретизации их связей. Кроме того, морфологические таблицы строят в предположении, что множества $\mathbf{R}_i$ взаимно независимы, т.е. состав способов реализации *i*-й функции не меняется при изменении значений других функций. Очевидно, что предположение о взаимной независимости множеств $\mathbf{R}_i$ оправдано лишь в сравнительно простых структурах. Последний недостаток устраняется путем обобщения метода морфологических таблиц – при использовании метода альтернативных (И-ИЛИ) графов.

Альтернативные графы. Любую морфологическую таблицу можно представить в виде дерева (рис. 4.12). На рисунке функции представлены вершинами И (темные кружки), значения функций — вершинами ИЛИ (светлые кружки). Очевидно, что таблица представляет множество однотипных объектов, поскольку все они характеризуются одним и тем же множеством функций.

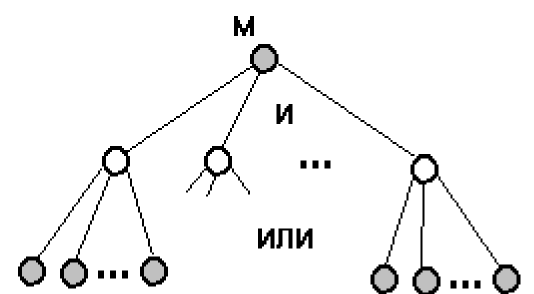


Рис. 4.12. Дерево, соответствующее морфологической таблтце

Для разнотипных объектов применяют многоярусные альтернативные графы. Например, на рис. 4.13 показан двухъярусный граф, в котором для разных типов объектов предусмотрены разные подмножества функций.

Если допустить некоторую избыточность при изображении И-ИЛИ- графа, то его можно превратить в И-ИЛИ-дерево, что ведет к определенным удобствам.

Очевидно, что И-ИЛИ-дерево можно представить как совокупность морфологических таблиц. Каждая И вершина дерева соответствует частной морфологической таблице, т.е. множеству функций так, что *i*-я выходящая ветвь отображает *i*-ю функцию. Каждая ИЛИ вершина, инцидентная *i*-й ветви, соответствует множеству вариантов реализации *i*-й функции, при этом *j*-я исходящая из ИЛИ вершины ветвь отображает *j*-й вариант реализации.

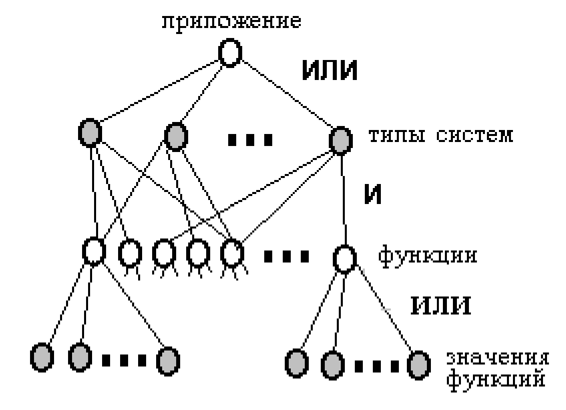


Рис. 4.13. И-ИЛИ-граф

Алгоритмизация синтеза на базе И-ИЛИ-деревьев требует введения правил выбора альтернатив в каждой вершине ИЛИ. Эти правила чаще всего имеют эвристический характер, связаны с требованиями ТЗ, могут отражать запреты на сочетания определенных компонентов структур.
Трудности эффективного решения задачи существенно возрастают при наличии ограничений, типичными среди которых являются ограничения на совместимость способов реализации разных функций, т.е. ограничения вида

$$C_{ij} \textbf{ and } C_{pq} = \text{false}, \qquad (4.29)$$

где C_{ij} = true, если в оцениваемый вариант вошел элемент $Э_{ij}$, иначе C_{ij} = false. Условие (4.29) означает, что в допустимую структуру не могут входить одновременно элементы $Э_{ij}$ и $Э_{pq}$. Совокупность ограничений типа (4.29) можно представить как систему логических уравнений с неизвестными C_{ij}. Тогда задачу синтеза можно решать эволюционными методами, если предварительно или одновременно с ней решать систему логических уравнений (задачу о выполнимости).

Исчисления. Очевидно, что в большинстве случаев структурного синтеза вместо нереализуемого явного представления всего множества проектных решений задают множество элементов и совокупность правил объединения этих элементов в допустимые структуры (проектные решения).

Эти множества элементов и правил часто представляют в виде *формальной системы (исчисления)*, т.е. задача синтеза имеет вид

ЗС = <**Э; НТ; АК; П**>,

где **Э** — алфавит исчисления (алфавит представлен базовыми элементами, из которых синтезируется структура); **НТ** — множество букв, не совпадающих с буквами алфавита **Э** и служащих для обозначения переменных; **АК** — множество аксиом исчисления, под которыми понимаются задаваемые исходные формулы (слова) в алфавите **Э** (например, соответствия функций и элементов); **П** — множество правил вывода новых формул в алфавите **Э** из аксиом и ранее выведенных корректных формул. Каждую формулу можно интерпретировать как некоторую структуру, поэтому синтез — это процесс вывода формулы, удовлетворяющей исходным требованиям и ограничениям.

Другие примеры компактного задания множества альтернатив **А** через множества **Э** и **П** связан с использованием систем искусственного интеллекта, в которых **Э** есть база данных, **П** — база знаний, или эволюционных методов, в которых **Э** — также база данных, **П** — множество эвристик, последовательность применения которых определяется эволюционными и генетическими принципами.

4.4. Методы структурного синтеза в САПР

Системы искусственного интеллекта. В теории интеллектуальных систем синтез реализуется с помощью экспертных систем (ЭС)

ЭС = <БД, БЗ, И>,

где БД — база данных, включающая сведения о базовых элементах; БЗ — база знаний, содержащая правила конструирования вариантов структуры; И — интерпретатор, устанавливающий последовательность применения правил из БЗ. Системы искусственного интеллекта (СИИ) основаны на знаниях, отделенных от процедурной части программ и представленных в одной из характерных форм. Такими формами могут быть продукции, фреймы, семантические сети. Реально функционирующие в современных САПР системы с базами знаний чаще всего относятся к классу ЭС.

Продукция представляет собой правило типа “если A, то B”, где A — условие, а B — действие или следствие, активизируемое при истинности A. Продукционная БЗ содержит совокупность правил, описывающих определенную предметную область.

Фрейм — структура данных, в которой в определенном порядке представлены сведения о свойствах описываемого объекта. Типичный вид фрейма:

$<$имя фрейма; $x_1 = p_1$; $x_2 = p_2$;...; $x_N = p_N$; $q_1, q_2,...q_M>$,

где x_i — имя i-го атрибута, p_i — его значение, q_i — ссылка на другой фрейм или некоторую обслуживающую процедуру. В качестве p_i можно использовать имя другого (вложенного) фрейма, описывая тем самым иерархические структуры фреймов.

Семантическая сеть — форма представления знаний в виде совокупности понятий и явно выраженных отношений между ними в некоторой предметной области. Семантическую сеть удобно представлять в виде графа, в котором вершины отображают понятия, а ребра или дуги — отношения между ними. В качестве вершин сети можно использовать фреймы или продукции.

Экспертная система является типичной системой искусственного интеллекта, в которой БЗ содержит сведения, полученные от людей-экспертов в конкретной предметной области. Трудности формализации процедур структурного синтеза привели к популярности применения экспертных систем в САПР, поскольку в них вместо выполнения синтеза на базе формальных математических методов осуществляется синтез на основе опыта и неформальных рекомендаций, полученных от экспертов.

Дискретное математическое программирование. Выбор метода поиска решения — вторая проблема после формализации задачи. Если при формализации все управляемые параметры удалось представить в числовом виде, то можно попытаться применить известные методы ДМП.

Задача ДМП определяется следующим образом:

$$\underset{\mathbf{X}\in\mathbf{D}}{\mathbf{extr}}\, F(\mathbf{X}), \qquad (4.30)$$

$$\mathbf{D} = \{ \mathbf{X} \mid \mathbf{W}(\mathbf{X}) > 0, \mathbf{Z}(\mathbf{X}) = 0 \},$$

где $F(\mathbf{X})$ — целевая функция; $\mathbf{W(X), Z (X)}$ — вектор-функции, связанные с представленными в ТЗ требованиями и ограничениями**;** $\mathbf{D}$ — дискретное множество. В полученной модели, во-первых, каждый элемент множества рассматриваемых законченных структур должен иметь уникальное сочетание значений некоторого множества числовых параметров, вектор которых обозначим $\mathbf{X}$. Во-вторых, необходимо существование одной или нескольких функций $\mathbf{\Phi(X)}$**,** значения которых могут служить исчерпывающей оценкой соответствия структуры предъявляемым требованиям. В-третьих, функции $\mathbf{\Phi(X)}$ должны отражать внутренне присущие данному классу объектов свойства, что обеспечит возможность использования $\mathbf{\Phi(X)}$ в качестве не только средств оценки достигнутого при поиске успеха, но и средств указания перспективных направлений продолжения поиска. Эти условия выполнимы далеко не всегда, что и обусловливает трудности формализации задач структурного синтеза.

Однако наличие формулировки (4.30) еще не означает, что удастся подобрать метод (алгоритм) решения задачи (4.30) с приемлемыми затратами вычислительных ресурсов. Другими словами, применение точных методов математического программирования вызывает непреодолимые трудности в большинстве случаев практических задач типичного размера из-за их принадлежности к классу **NP**-трудных задач. Поэтому лидирующее положение среди методов решения задачи (4.30) занимают приближенные методы, в частности, декомпозиционные методы, отражающие принципы блочно-иерархического проектирования сложных объектов. Декомпозиционные методы основаны на выделении ряда иерархических уровней, на каждом из которых решаются задачи приемлемого размера.

Основу большой группы математических методов, выражающих стремление к сокращению перебора, составляют операции разделения множества вариантов на подмножества и отсечения неперспективных подмножеств. Эти методы объединяются под названием *метода ветвей и границ*. Основная разновидность метода ветвей и границ относится к точным методам решения комбинаторных задач. Рассмотрим эту разновидность.

Пусть имеется множество решений $\mathbf{M}$, в котором нужно выбрать оптимальный по критерию $F(\mathbf{X}_j)$ вариант, где $\mathbf{X}_j$ — вектор параметров варианта $m_j \in \mathbf{M}$; пусть также имеется алгоритм для вычисления нижней границы $L(\mathbf{M}_k)$ критерия $F(\mathbf{X}_j)$ в любом подмножестве $\mathbf{M}_k$ множества $\mathbf{M}$, т.е. такого значения $L(\mathbf{M}_k)$, что $F(\mathbf{X}_j) \geq L(\mathbf{M}_k)$ при любом j (подразумевается минимизация $F(\mathbf{X})$). Тогда основная схема решения задач в соответствии с методом ветвей и границ содержит следующие процедуры: 1) в качестве $\mathbf{M}_k$ принимаем все множество $\mathbf{M}$; 2) ветвление: разбиение $\mathbf{M}_k$ на несколько подмножеств $\mathbf{M}_q$; 3) вычисление нижних границ $L(\mathbf{M}_q)$ в подмножествах $\mathbf{M}_q$; 4) выбор в качестве $\mathbf{M}_k$ подмножества $\mathbf{M}_p$ с минимальным значением нижней границы критерия (среди всех подмножеств, имеющихся на данном этапе вычислений), сведения об остальных подмножествах $\mathbf{M}_q$ и их нижних границах сохраняются в отдельном списке; 5) если $|\mathbf{M}_k| > 1$, то переход к процедуре 2, иначе одноэлементное множество $\mathbf{M}_k$ есть решение.

Метод ветвей и границ в случае точного вычисления нижних границ относится к точным методам решения задач выбора и потому в неблагоприятных ситуациях может приводить к экспоненциальной временной сложности. Однако метод часто используют как приближенный, поскольку можно применять приближенные алгоритмы вычисления нижних границ.

Среди других приближенных методов решения задачи ДМП отметим *метод локальной оптимизации*. Так как пространство $\mathbf{D}$ метризовано, то можно использовать понятие a-окрестности $\mathbf{S}_a(\mathbf{X}_k)$ текущей точки поиска $\mathbf{X}_k$. Вместо перебора точек во всем пространстве $\mathbf{D}$ осуществляется перебор точек только в $\mathbf{S}_a(\mathbf{X}_k)$. Если $F(\mathbf{X}_j) \geq F(\mathbf{X}_k)$ для всех $\mathbf{X}_j \in \mathbf{S}_a(\mathbf{X}_k)$, то считается, что найден локальный минимум целевой функции в точке $\mathbf{X}_k$. В противном случае точку $\mathbf{X}_q$, в которой достигается минимум $F(\mathbf{X})$ в $\mathbf{S}_a(\mathbf{X}_k)$, принимают в качестве новой текущей точки поиска.

Элементы теории сложности. В теории сложности выделяют массовые и индивидуальные задачи. Первые из них сформулированы в общем виде, вторые представлены с конкретными числовыми значениями исходных данных. Исследования сложности проводятся в отношении массовых задач и получаемые выводы, как правило, относятся к наихудшему случаю — к наиболее неблагоприятному возможному сочетанию исходных данных.

Цель исследований — установление вида зависимости объема Q требуемых вычислений от размера задачи N. Объем вычислений может определяться числом арифметических и логических операций или затратами процессорного времени ЭВМ с заданной производительностью. Размер задачи в общем случае связывают с объемом описания задачи, но в приложениях понятие размера легко наполняется более конкретным содержанием.

Далее, в теории сложности задач выбора вводят понятие эффективных и неэффективных алгоритмов. К *эффективным* относят алгоритмы с полиномиальной зависимостью Q от N, например, алгоритмы с функцией $Q(N)$ линейной, квадратичной, кубической и др. Для *неэффективных* алгоритмов характерна экспоненциальная зависимость $Q(N)$.

Важность проведения резкой границы между полиномиальными и экспоненциальными алгоритмами вытекает из сопоставления числовых примеров роста допустимого размера задачи с увеличением быстродействия Б используемых ЭВМ (табл. 4.1, в которой указаны размеры задач, решаемых за одно и то же время T на ЭВМ с быстродействием $Б_i$ при различных зависимостях сложности Q от размера N). Эти примеры показывают, что выбирая ЭВМ в K раз более быстродействующую, получаем увеличение размера решаемых задач при линейных алгоритмах в K раз, при квадратичных алгоритмах в $K^{1/2}$ раз и т.д.

Таблица 4.1

$Q(N)$	$Б_1$	$Б_2 = 100\ Б_1$	$Б_3 = 1000\ Б_1$
N	N_1	$100\ N_1$	$1000\ N_1$
N^2	N_2	$10\ N_2$	$31.6\ N_2$
N^3	N_3	$4.64\ N_3$	$10\ N_3$
2^N	N_4	$6.64{+}N_4$	$9.97{+}N_4$

Иначе обстоит дело с неэффективными алгоритмами. Так, в случае сложности 2^N для одного и того же процессорного времени размер задачи увеличивается только на $\lg K$ / $\lg 2$ единиц. Следовательно, переходя от ЭВМ с Б = 1 Gflops к суперЭВМ с Б = 1 Tflops, можно увеличить размер решаемой задачи только на 10, что совершенно недостаточно для практических задач. Действительно, в таких задачах, как например, синтез тестов для БИС число входных двоичных переменных может составлять более 150 и поэтому полный перебор всех возможных проверяющих кодов потребует выполнения более 2^{150} вариантов моделирования схемы.

В теории сложности все комбинаторные задачи разделены на классы:

— класс неразрешимых задач, в который входят массовые задачи, решение которых полным перебором принципиально невозможно с точки зрения современных научных представлений; этот класс отделяется от других задач так называемым пределом Бреммермана, оцениваемым величиной $N = 10^{93}$; отметим, что реальный предел неразрешимости значительно ниже;

— класс **P**, к которому относятся задачи, для которых известны алгоритмы решения полиномиальной сложности;

— класс **NP**, включающий задачи, для которых можно за полиномиальное время проверить правильность решения, т.е. ответить на вопрос, удовлетворяет ли данное решение заданным условиям; очевидно, что **P** включено в **NP**, однако вопрос о совпадении этих классов пока остается открытым, хотя по-видимому на этот вопрос будет получен отрицательный ответ;

— класс **NP**-полных задач, характеризующийся следующими свойствами: 1) для этих задач неизвестны полиномиальные алгоритмы точного решения; 2) любые задачи внутри этого класса могут быть сведены одна к другой за полиномиальное время. Последнее означает, что если будет найден полиномиальный алгоритм для точного решения хотя бы одной **NP**-полной задачи, то за полиномиальное время можно будет решить любую задачу этого класса.

Из результатов теории сложности следуют важные практические рекомендации: 1) приступая к решению некоторой комбинаторной задачи, следует сначала проверить, не принадлежит ли она к

классу **NP**-полных задач, и если это так, то не следует тратить усилия на разработку алгоритмов и программ точного решения; 2) отсутствие эффективных алгоритмов точного решения массовой задачи выбора отнюдь не означает невозможности эффективного решения индивидуальных задач из класса **NP**-полных или невозможности получения приближенного решения по эвристическим алгоритмам за полиномиальное время.

Эволюционные методы. *Эволюционные методы* (ЭМ) предназначены для поиска предпочтительных решений и основаны на статистическом подходе к исследованию ситуаций и итерационном приближении к искомому состоянию систем.

В отличие от точных методов математического программирования ЭМ позволяют находить решения, близкие к оптимальным, за приемлемое время, а в отличие от известных эвристических методов оптимизации характеризуются существенно меньшей зависимостью от особенностей приложения (т.е. более универсальны) и в большинстве случаев обеспечивают лучшую степень приближения к оптимальному решению. Универсальность ЭМ определяется также применимостью к задачам с неметризуемым пространством управляемых переменных (т.е. среди управляемых переменных могут быть и лингвистические).

Важнейшим частным случаем ЭМ являются *генетические методы и алгоритмы*. Генетические алгоритмы (ГА) основаны на поиске лучших решений с помощью наследования и усиления полезных свойств множества объектов определенного приложения в процессе имитации их эволюции.

Свойства объектов представлены значениями параметров, объединяемыми в запись, называемую в ЭМ *хромосомой*. В ГА оперируют хромосомами, относящимися к множеству объектов — *популяции*. Имитация генетических принципов — вероятностный выбор родителей среди членов популяции, скрещивание их хромосом, отбор потомков для включения в новые поколения объектов на основе оценки целевой функции — ведет к эволюционному улучшению значений целевой функции (функции полезности) от поколения к поколению.

Среди ЭМ находят применение также методы, которые в отличие от ГА оперируют не множеством хромосом, а единственной хромосомой. Так, метод дискретного *локального поиска (*его англоязычное название *Hillclimbing*) основан на случайном изменении отдельных параметров (т.е. значений полей в записи или, другими словами, значений генов в хромосоме). Такие изменения называют *мутациями.* После очередной мутации оценивают значение *функции полезности F* (Fitness Function) и результат мутации сохраняется в хромосоме только, если F улучшилась. В другом ЭМ под названием “*Моделирование отжига*” (Simulated Annealing) результат мутации сохраняется с некоторой вероятностью, зависящей от полученного значения F.

Постановка задачи поиска оптимальных решений с помощью генетических алгоритмов. Для применения ГА необходимо:

1) выделить совокупность свойств объекта, характеризуемых внутренними параметрами и влияющих на его полезность, т.е. выделить множество управляемых параметров $\mathbf{X} = (x_1,x_2,...x_n)$; среди x_i могут быть величины различных типов (real, integer, Boolean, enumeration). Наличие нечисловых величин (enumeration) обусловливает возможность решения задач не только параметрической, но и структурной оптимизации;

2) сформулировать количественную оценку полезности вариантов объекта — функцию полезности F. Если в исходном виде задача многокритериальна, то такая формулировка означает выбор скалярного (обобщенного) критерия;

3) Разработать математическую модель объекта, представляющую собой алгоритм вычисления F для заданного вектора $\mathbf{X}$;

4) Представить вектор $\mathbf{X}$ в форме хромосомы — записи следующего вида

X_1	X_2	X_3		X_n

В ГА используется следующая терминология:

ген — управляемый параметр x_i;

аллель — значение гена;

локус (позиция) — позиция, занимаемая геном в хромосоме;

генотип — экземпляр хромосомы, генотип представляет совокупность внутренних параметров проектируемого с помощью ГА объекта;

генофонд — множество всех возможных генотипов;

функция полезности (приспособленности) F — целевая функция;

фенотип — совокупность генотипа и соответствующего значения F, под фенотипом часто понимают совокупность выходных параметров синтезируемого с помощью ГА объекта.

Простой генетический алгоритм. Вычислительный процесс начинается с генерации исходного поколения — множества, включающего N хромосом, N — размер популяции. Генерация выполняется случайным выбором аллелей каждого гена.

Далее организуется циклический процесс смены поколений:

```
for (k=0; k<G; k++)
{ for (j=0; j<N; j++)
  { Выбор родительской пары хромосом;
    Кроссовер;
    Мутации;
    Оценка функции полезности F потомков;
    Селекция;
  }
  Замена текущего поколения новым;
}
```

Для каждого витка внешнего цикла генетического алгоритма выполняется внутренний цикл, на котором формируются экземпляры нового (следующего за текущим) поколения. Во внутреннем цикле повторяются операторы выбора родителей, кроссовера родительских хромосом, мутации, оценки приспособленности потомков, селекции хромосом для включения в очередное поколение.

Рассмотрим алгоритмы выполнения операторов в простом генетическом алгоритме.

Выбор родителей. Этот оператор имитирует естественный отбор, если отбор в родительскую пару хромосом с лучшими значениями функции полезности F более вероятен. Например, пусть F требуется минимизировать. Тогда вероятность P_i выбора родителя с хромосомой C_i можно рассчитать по формуле

$$P_i = (F_{max} - F_i) \Big/ \sum_{j=1}^{N} (F_{max} - F_j) \qquad (4.31)$$

где F_{max} — наихудшее значение целевой функции F среди экземпляров (членов) текущего поколения, F_i — значение целевой функции i-го экземпляра.

Правило (4.31) называют *правилом колеса рулетки*. Если в колесе рулетки выделить секторы, пропорциональные значениям F_{max}-F_i, то вероятности попадания в них суть P_i, определяемые в соответствии с (4.31).

Пример. Пусть N=4, значения F_i и P_i приведены в табл. 4.2.

Таблица 4.2

i	F_i	F_{max}-F_i	P_i
1	2	5	0,5
2	7	0	0
3	6	1	0,1
4	3	4	0,4

Кроссовер (скрещивание). Кроссовер, иногда называемый кроссинговером, заключается в передаче участков генов от родителей к потомкам. При простом (одноточечном) кроссовере хромосомы родителей разрываются в некоторой позиции, одинаковой для обоих родителей, выбор места разрыва равновероятен, далее происходит рекомбинация образующихся частей родительских хромосом, как это показано в табл. 4.3, где разрыв подразумевается между пятым и шестым локусами.

Таблица 4.3

Хромосома	Гены							
родителя **A**	**f**	**a**	**c**	**d**	**g**	**k**	**v**	**e**
родителя **B**	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*
потомка **C**	**f**	**a**	**c**	**d**	**g**	*f*	*g*	*h*
потомка **D**	*a*	*b*	*c*	*d*	*e*	**k**	**v**	**e**

Мутации. Оператор мутации выполняется с некоторой вероятностью $P_м$, т.е. с вероятностью $P_м$ происходит замена аллеля случайным значением, выбираемым с равной вероятностью в области определения гена. Именно благодаря мутациям расширяется область генетического поиска.

Селекция. После каждого акта генерации пары потомков в новое поколение включается лучший экземпляр пары.

Внутренний цикл заканчивается, когда число экземпляров нового поколения станет равным *N*. Количество повторений *G* внешнего цикла чаще всего определяется автоматически по появлению признаков вырождения (стагнации) популяции, но с условием не превышения заданного лимита машинного времени.

Разновидности генетических операторов. Возможны отклонения от представленной выше в простом генетическом алгоритме схемы вычислений.

Кроссовер. Во-первых, допустимы схемы многоточечного кроссовера.

Во-вторых, отметим ситуации, когда на состав аллелей наложены некоторые дополнительные условия. Например, пусть в задаче разбиения графа число вершин в подграфах $\mathbf{A}_1$ и $\mathbf{A}_2$ должно быть N_1 и N_2 и пусть *k*-й аллель, равный 1, означает, что вершина *k* попадает в $\mathbf{A}_1$, если же *k*-й аллель равен 0, то в $\mathbf{A}_2$. Очевидно, что число единиц в хромосоме должно равняться N_1, число нулей — N_2. Тогда при рекомбинации левый участок хромосомы берется от одного из родителей без изменений, а в правом участке (от другого родителя) нужно согласовать число единиц с N_1 тем или иным способом.

Один из способов — метод PMX (Partially Matched Crossover). Для иллюстрации PMX рассмотрим пример двухточечного кроссовера в задаче, когда в хромосоме должны присутствовать, причем только по одному разу, все значения генов из заданного набора. Пусть в примере этот набор включает числа от 1 до 9.

Таблица 4.4

1	2		3	4	5		6	7	8	9
3	7		1	9	2		4	8	6	5
1	2		1	9	2		6	7	8	9
1	2		3	9	5		6	7	8	4

В табл. 4.4 первые две строки представляют родительские хромосомы. Третья строка содержит хромосому одного из потомков, сгенерированного в результате применения двухточечного кроссовера (после второго и пятого локусов). Полученная хромосома не относится к числу допустимых, так как в ней значения генов 1, 2 и 9 встречаются дважды, а значения 3, 4 и 5 отсутствуют. Четвертая строка показывает результат применения PMX. В этом методе выделяются сопряженные пары аллелей в одноименных локусах одной из рекомбинируемых частей. В нашем примере это пары (3 и 1), (4 и 9), (5 и 2). Хромосома потомка просматривается слева направо; если повторно встречается некоторое значение, оно заменяется на сопряженное значение. Так, в примере в локусах 3, 5 и 9 повторно встречаю-

щиеся аллели 1, 2 и 9 последовательно заменяются на значения 3, 5 и 4.

Мутации. Бывают точечными (в одном гене), макромутациями (в нескольких генах) и хромосомными (появление новой хромосомы). Обычно вероятность появления мутации указывается среди исходных данных. Но возможно автоматическое регулирование числа мутаций при их реализации только в ситуациях, когда родительские хромосомы различаются не более чем в K генах.

Селекция. После определения и положительной оценки потомка он может быть сразу же включен в текущую популяцию вместо худшего из своих родителей, при этом из алгоритма исключается внешний цикл (что однако не означает сокращения общего объема вычислений).

Другой вариант селекции — отбор после каждой операции скрещивания двух лучших экземпляров среди двух потомков и двух родителей.

Часто член популяции с минимальным (лучшим) значением целевой функции принудительно включается в новое поколение, что гарантирует наследование приобретенных этим членом положительных свойств. Такой подход называют *элитизмом*. Обычно элитизм способствует более быстрой сходимости к локальному экстремуму, однако в многоэкстремальной ситуации ограничивает возможности попадания в окрестности других локальных экстремумов.

Примечание. Хромосому $\mathbf{X}^*$ будем называть точкой локального минимума, если $F(\mathbf{X}^*)<F(\mathbf{X}_i)$ для всех хромосом $\mathbf{X}_i$, отличающихся от $\mathbf{X}^*$ значением единственного гена, где $F(\mathbf{X})$ — значение функции полезности в точке $\mathbf{X}$.

Следующий вариант селекции — отбор N экземпляров среди членов репродукционной группы, которая составляется из родителей, потомков и мутантов, удовлетворяющих условию $F_i < t$, где t — пороговое значение функции полезности. Порог может быть равен или среднему значению F в текущем поколении, или значению F особи, занимающей определенное порядковое место. При этом мягкая схема отбора — в новое поколение включаются N лучших представителей репродукционной группы. Жесткая схема отбора — в новое поколение экземпляры включаются с вероятностью q_i:

$$q_i = (F_{\max}-F_i) / \sum_{j=1}^{N_r} (F_{\max} - F_j)$$

где N_r — размер репродукционной группы.

Переупорядочение. Кроме перечисленных основных операторов, находят применение некоторые дополнительные. К их числу относится оператор переупорядочения генов — изменения их распределения по локусам.

Назначение переупорядочения связано со свойством, носящим название эпистасис. *Эпистасис* имеет место, если функция полезности зависит не только от значений генов (аллелей), но и от их позиционирования. Наличие эпистасиса говорит о нелинейности целевой функции и существенно усложняет решение задач. Действительно, если некоторые аллели двух генов оказывают определенное положительное влияние на целевую функцию, образуя некоторую связку (схему), но вследствие эпистасиса при разрыве связки эти аллели оказывают уже противоположное влияние на функцию полезности, то разрывать такие схемы не следует. А это означает, что связанные эпистасисом гены желательно располагать близко друг к другу, т.е. при небольших длинах схем. Оператор переупорядочения помогает автоматически “нащупать” такие совокупности генов (они называются хромосомными блоками или building blocks) и разместить их в близких локусах.

Генетический метод комбинирования эвристик. Возможны два подхода к формированию хромосом.

Первый из них основан на использовании в качестве генов проектных параметров. Например, в задаче размещения микросхем на плате локусы соответствуют посадочным местам на плате, а генами являются номера (имена) микросхем. Другими словами, значением k-го гена будет номер микросхемы в k-й позиции.

Во втором подходе генами являются не сами проектные параметры, а номера эвристик, используемых для определения проектных параметров. Так, для задачи размещения можно применять несколько эвристик. По одной из них в очередное посадочное место нужно помещать микросхему, имеющую наибольшее число связей с уже размещенными микросхемами, по другой — микросхему с минимальным числом связей с еще не размещенными микросхемами и т.д. Генетический поиск в этом

случае есть поиск последовательности эвристик, обеспечивающей оптимальный вариант размещения.

Второй подход получил название — *метод комбинирования эвристик*. Этот метод оказывается предпочтительным во многих случаях. Например, в задачах синтеза расписаний распределяется заданное множество работ во времени и между обслуживающими устройствами — серверами, т.е. проектными параметрами для каждой работы будут номер сервера и порядковый номер в очереди на обслуживание. Пусть N — число работ, M — число серверов. Если гены соответствуют номерам работ, то в первом подходе в хромосоме нужно иметь $2N$ генов и общее число отличающихся друг от друга хромосом W заметно превышает наибольшее из чисел $N!$ и M^N.

Согласно методу комбинирования эвристик, число генов в хромосоме в два раза меньше, чем в первом подходе, и равно N. Поэтому если число используемых эвристик равно K, то мощность множества возможных хромосом уже несравнимо меньше, а именно

$$W = K^N.$$

Очевидно, что меньший размер хромосомы ведет к лучшей вычислительной эффективности, а меньшее значение W позволяет быстрее найти окрестности искомого экстремума. Кроме того, в методе комбинирования эвристик все хромосомы, генерируемые при кроссовере, будут допустимыми. В то же время при применении обычных генетических методов необходимо использовать процедуры типа PMX для корректировки генов, относящихся к номерам в очереди на обслуживание, что также снижает эффективность поиска.

Упражнения и вопросы для самоконтроля

1. Дайте формулировку задачи математического программирования.

2. В чем заключаются трудности решения многокритериальных задач оптимизации?

3. Что такое “множество Парето”?

4. Для функции, заданной своими линиями равного уровня (рис. 4.14), постройте траектории поиска методами конфигураций, деформируемого многогранника, наискорейшего спуска из исходной точки $\mathbf{X}_0$.

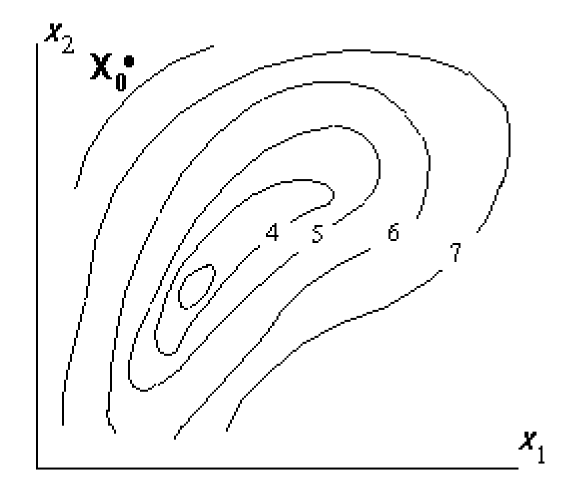


Рис. 4.14. Пример для построения траекторий поиска

5. Как Вы считаете, можно ли применять метод проекции градиента для решения задач оптимизации с ограничениями типа неравенств?

6. Что такое “овражная целевая функция”? Приведите пример такой функции для двумерного случая в виде совокупности линий равного уровня.

7. Какие свойства характеризуют класс **NP**-полных задач?

8. Морфологическая таблица содержит 8 строк и 24 столбца. Сколько различных вариантов структуры представляет данная таблица?

9. Приведите пример И-ИЛИ графа для некоторого знакомого Вам приложения.

10. Приведите примеры продукций из знакомого Вам приложения.

11. Дайте предложения по постановке задачи компоновки модулей в блоки для ее решения генетическими методами. Какова структура хромосомы?



Глава 5 Системные среды и программно-методические комплексы САПР

5.1. Функции сетевого программного обеспечения

Функции и характеристики сетевых операционных систем. В ПО АС принято выделять общесистемное ПО, системные среды и прикладное ПО.

К общесистемному ПО относят операционные системы (ОС) используемых ЭВМ и вычислительных систем и сетевое ПО типовых телекоммуникационных услуг.

Различают ОС со встроенными сетевыми функциями и оболочки над локальными ОС. В соответствии с другим признаком классификации сетевые ОС подразделяют на одноранговые и функционально несимметричные (ОС для систем клиент-сервер).

Основные функции сетевой ОС:

— управление каталогами и файлами;
— управление ресурсами;
— коммуникационные функции;
— защита от несанкционированного доступа;
— обеспечение отказоустойчивости;
— управление сетью.

Управление каталогами и файлами является одной из первоочередных функций сетевой ОС, обслуживаемых специальной сетевой файловой подсистемой. Пользователь получает от этой подсистемы возможность обращаться к файлам, физически расположенным в сервере или в другой станции данных, применяя привычные для локальной работы языковые средства.

Управление ресурсами включает в себя функции запроса и предоставления ресурсов.

Коммуникационные функции обеспечивают адресацию, буферизацию, маршрутизацию сообщений.

Защита от несанкционированного доступа возможна на любом из следующих уровней: ограничение доступа в определенное время, и (или) для определенных станций, и (или) заданное число раз; ограничение совокупности доступных конкретному пользователю директорий; ограничение для конкретного пользователя списка возможных действий (например, только чтение файлов); пометка файлов символами типа “только чтение”, “скрытность при просмотре списка файлов”.

Отказоустойчивость определяется наличием у серверов автономных источников питания, отображением или дублированием информации в дисковых накопителях. Отображение заключается в хранении двух копий данных на двух дисках, подключенных к одному контроллеру, а дублирование означает подключение каждого из этих двух дисков к разным контроллерам. Сетевая ОС, реализующая дублирование дисков, обеспечивает более высокий уровень отказоустойчивости. Дальнейшее повышение отказоустойчивости связано с дублированием серверов.

Чем сложнее сеть, тем острее встают вопросы *управления сетью*. Основные функции управления сетью реализуются в ПО, поддерживающем протоколы управления такие, как ICMP и SNMP в стеке TCP/IP или протокол CMIP (Common Management Information Protocol) в семиуровневой модели ISO. Как рассмотрено выше, это ПО представлено менеджерами и агентами. Менеджер — прикладная программа, выдающая сетевые команды. Агенты доводят эти команды до исполнительных устройств и сигнализируют о событиях в состоянии устройств, они следят за трафиком и фиксируют аномалии, помогают восстановлению информации после сбоев, борются с вирусами и т.п.

В сетевых ОС обычно выделяют ядро, реализующее большинство из перечисленных функций и ряд дополнительных программ (служб), ориентированных на реализацию протоколов верхних уровней, организацию распределенных вычислений и т.п. К сетевому ПО относятся также драйверы сетевых плат, различные для разных типов ЛВС (Ethernet, TR, AppleTalk и др.).

В настоящее время выбор среди ОС происходит преимущественно между тремя основными операционными системами — UNIX, Windows NT, Novell Netware.

Областью применения ОС UNIX остаются крупные корпоративные сети со стеком протоколов TCP/IP. Отличительные свойства UNIX — высокая надежность, возможность легкого масштабирования сети.

Windows NT предназначена для работы в сетях клиент-сервер, ориентирована преимущественно на рабочие группы и средние по своим масштабам сети. ОС асимметрична — включает в себя серверную (Windows NT Server) и клиентскую (Windows NT Workstation) части.

Novell Netware пока сохраняет свои позиции в небольших сетях. Состоит из серверной части и оболочек Shell, размещаемых в клиентских узлах.

Системы распределенных вычислений. При выполнении проектных процедур с использованием более чем одного узла сети различают режимы удаленного узла и дистанционного управления (рис. 5.1).

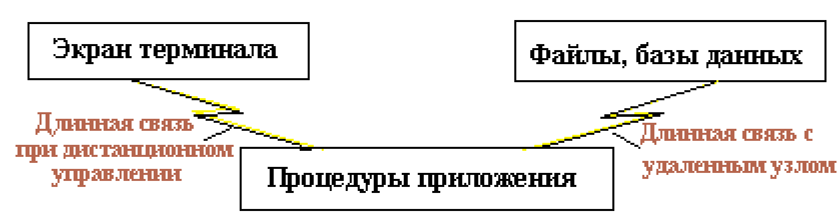


Рис. 5.1. Удаленный узел и дистанционное управление

В режиме *удаленного узла* основные процедуры приложения исполняются на терминальном узле. Связь с удаленным узлом используется для пересылки файлов. В большинстве случаев режим удаленного узла приводит к более заметной инерционности связи, чем режим дистанционного управления.

Дистанционное управление обеспечивает передачу клавишных команд в прямом направлении и экранных изображений (обычно лишь изменений в них) в сжатом виде в обратном направлении, поэтому задержки меньше.

Системы распределенных вычислений основаны на режиме дистанционного управления, при котором терминальный узел используется преимущественно для интерфейса с пользователем и передачи команд управления, а основные процедуры приложения исполняются на удаленном узле (сервере). Поэтому в сетях распределенных вычислений должны быть выделены серверы приложений.

Программное обеспечение организации распределенных вычислений называют ПО *промежуточного слоя* (Middleware). Современная организация распределенных вычислений в сетях Internet/Intranet основана на создании и использовании программных средств, которые могут работать в различных аппаратно-программных средах. Совокупность таких средств называют также *многоплатформенной распределенной средой* — МРС (Crossware).

Находят применение технологии распределенных вычислений RPC (Remote Procedure Call), ORB (Object Request Broker), DCE (Distributed Computing Environment), мониторы транзакций TPM (Transaction Processing Monitors) и др.

Средства RPC входят во многие системы сетевого ПО. RPC — процедурная блокирующая синхронная технология, предложенная фирмой Sun Microsystems. Вызов удаленных программ подобен вызову функций в языке С. При пересылках на основе транспортных протоколов TCP или UDP данные представляются в едином формате обмена. Синхронность и блокирование означают, что клиент, обратившись к серверу, для продолжения работы ждет ответа от сервера.

Для систем распределенных вычислений разработаны специальные языки, например для RPC — язык IDL (Interface Definition Language), который позволяет пользователю оперировать различными объектами безотносительно к их расположению в сети. На этом языке можно записывать обращения к серверам приложений.

Рассмотрим типичную схему реализации RPC.

Удаленная программа характеризуется атрибутами: имя узла, номер программы (часто номер означает совокупность программ определенного назначения), версия программы (версия — это идентификатор копии программы, например, версия — это время создания копии, копии создаются для использования в многопользовательском режиме), имя процедуры в программе.

Процедуры, которые пользователь собирается применять, необходимо зарегистрировать в узле-клиенте, т.е. указать имена узла, программы, процедуры.

Обращение по RPC — это обращение к сетевой программе Postmapper, находящейся в узле-клиенте. При обращении в запросе указываются процедура, аргумент, память под результат. Аргумент должен быть единственный, поэтому если аргументов много, то программист должен создать агрегат данных. Postmapper находит регистрационные данные и с помощью средств транспортного уровня устанавливает соединение и передает запрос серверу. В сервере имеется диспетчер, который находит ис-

полнителя запроса. В ответе сервера содержатся результаты выполнения процедуры.

ORB — технология объектно-ориентированного подхода, базирующаяся на спецификациях CORBA. Спецификации CORBA устанавливают способы использования удаленных объектов (серверных компонентов) в клиентских программах. Взаимодействие клиента с сервером происходит с помощью программы-посредника (брокера) ORB. В случае применения ORB (в отличие от RPC) хранить сведения о расположении серверных объектов в узле-клиенте не нужно, достаточно знать расположение в сети брокера ORB. Поэтому доступ пользователя к различным объектам (программам, данным, принтерам и т.п.) существенно упрощен. Брокер должен определять, в каком месте сети находится запрашиваемый ресурс и инициализировать серверную программу. После этого клиент может направлять запрос в серверный узел, а после выполнения запроса сервер будет возвращать результаты пользователю.

Для описания интерфейсов распределенных объектов используют язык IDL, предложенный в CORBA. Этот язык отличается от языка IDL технологии RPC, в нем имеются средства описания интерфейсов, но нет средств описания операций.

При использовании ORB может увеличиться нагрузка на сеть, однако имеется и ряд преимуществ: обеспечивается взаимодействие разных платформ, не требуется дублирования прикладных программ во многих узлах, упрощается программирование сетевых приложений и поддержка мультимедиа.

В CORBA создан протокол IIOP (Internet Inter-ORB Protocol), который обеспечивает взаимодействие между брокерами разных производителей.

Мониторы транзакций отличаются от RPC наличием готовых процедур обработки транзакций (в том числе отката транзакций), что упрощает работу программистов. Принимая запросы от клиентов и мультиплексируя их, монитор транзакций избавляет от необходимости создавать для каждого клиента отдельное соединение с БД. Мониторы транзакций могут оптимально распределять нагрузку на серверы, выполнять автоматическое восстановление после сбоя и перезапуск системы.

DCE разработана консорциумом OSF (Open Software Foundation). Она не противопоставляется другим технологиям (RPC, ORB), а является средой для их использования, например, в одной из реализаций DCE пакет Encina есть монитор транзакций, а пакет Orbix ORB представляет собой технологию ORB.

В DCE возможны одно- или многоячеечная структуры сети. Выделение ячеек производится по функциональным, а не по территориальным признакам. В каждой ячейке должен быть главный сервер данных и возможно несколько дополнительных серверов с копиями содержимого главного сервера, причем доступ к дополнительным серверам разрешен только для чтения. Обновление данных осуществляется исключительно через главный сервер. Ячейка может занимать значительную территорию, главный сервер размещается вблизи от центра ячейки, дополнительные серверы — по периферии.

К функциям DCE относятся распределение вычислений по технологии RPC; распараллеливание вычислений (но программист сам проектирует параллельный процесс); защита данных; синхронизация (согласование времени); поддержка распределенной файловой системы.

Работая в DCE, пользователь дополнительно к своей прикладной программе пишет IDL-файл, в котором указывает свое имя, требуемые операции и типы данных. IDL-компилятор на основе этого файла создает три модуля: клиентский стаб (Cl), серверный стаб (Sr), головной файл (Hd). Cl содержит вызовы процедур, Sr — обращения к базе процедур, Hd устанавливает связь между стабами.

Определение нужного сервера в DCE либо происходит автоматически с помощью ORB, либо возлагается на программиста, как в RPC .

Прикладные протоколы и телекоммуникационные информационные услуги. Основные услуги телекоммуникационных технологий — электронная почта, передача файлов, телеконференции, справочные службы (доски объявлений), видеоконференции, доступ к информационным ресурсам (информационным базам) сетевых серверов и др. Эти услуги обеспечиваются соответствующими прикладными протоколами.

Среди прикладных протоколов наиболее известны протоколы, связанные с Internet, и протоколы ISO-IP (ISO 8473), относящиеся к семиуровневой модели открытых систем. К важным прикладным протоколам Internet относятся следующие:

Telnet — протокол эмуляции терминала, или, другими словами, протокол реализации дистанци-

онного управления, он используется для подключения клиента к серверу при их размещении на разных компьютерах, пользователь через свой терминал имеет доступ к компьютеру-серверу;

FTP — протокол файлового обмена (реализуется режим удаленного узла), клиент может запрашивать и получать файлы с сервера, адрес которого указан в запросе;

HTTP (Hypertext Transmission Protocol) — протокол для связи Web-серверов и Web-клиентов;

SMTP, IMAP, POP3 — протоколы электронной почты;

SNMP — протокол управления сетью.

Указанные протоколы поддерживаются с помощью соответствующего ПО. Как правило, прикладной протокол реализуется серверной и клиентской программами. Клиентская программа запрашивает информационную услугу, серверная программа выполняет запрос. Для Telnet, FTP, SMTP на серверной стороне выделены фиксированные номера протокольных портов.

Электронная почта — средство обмена сообщениями по электронным коммуникациям (в режиме off-line). Посылка сообщения осуществляется по инициативе отправителя. Можно пересылать текстовые сообщения и архивированные файлы. В последних могут содержаться данные (например, тексты программ, графические данные) в различных форматах.

На ЭВМ пользователя должна быть установлена программа-клиент, поддерживающая функции создания, передачи и приема сообщений. На почтовом сервере, выделяемом в корпоративной или локальной сети, организуется промежуточное хранение поступающих сообщений. Связь индивидуальных пользователей с почтовым сервером осуществляется по протоколам IMAP или POP3.

В территориальных сетях почтовые сообщения проходят через ряд промежуточных федеральных или региональных узлов. В таких узлах устанавливают ПО (так называемый агент передачи сообщений), выполняющее функции сортировки и маршрутизации сообщений.

Разработан ряд альтернативных протоколов электронной почты для прикладного уровня. Расширение числа возможных кодировок и форматов данных по сравнению с SMTP сделано в MIME (Multipurpose Internet Mail Extensions). Применение MIME упрощает пересылку графических и звуковых файлов, реализацию шифрования и электронной подписи.

Примерами программ могут служить Lotus cc: mail, Microsoft Mail, Outlook Express и др.. Они позволяют посылать сообщения индивидуальному пользователю, на доску объявлений, последовательный просмотр несколькими исполнителями с возможностями коррекции сообщения; осуществляют поиск сообщений, пришедших в почтовый сервер, по контексту, по адресу, по времени отправки.

В настоящее время при разработке многих программных систем предусматривают интерфейс со средствами электронной почты, клиентские программы E-mail стараются включать в Web-браузеры сети Internet, а также во многие прикладные программные системы САПР, АСУ, документооборота.

Письма в E-mail состоят из заголовка и тела (текста). В заголовке указывается кому предназначено письмо, от кого оно поступило, кому посланы копии, дата отправки, указатель ключа, по которому пользователь может определить ключ для декодирования текста. В протоколе IMAP (Internet Message Access Protocol) сначала клиенту передается заголовок, а текст остается на сервере, затем пользователь при желании может получить и весь текст. В протоколе POP3 при обращении к почтовому серверу на клиентский узел переписывается все сообщение.

Файловый обмен — доступ к файлам, распределенным по различным компьютерам. Доступ возможен в режимах off-line и on-line. В режиме off-line посылается запрос к FTP-серверу, сервер формирует и посылает ответ на запрос. В режиме on-line осуществляется интерактивный просмотр каталогов FTP-сервера, выбор и передача нужных файлов. На ЭВМ пользователя устанавливается FTP-клиент.

При запросе файла по протоколу FTP пользователь должен знать, где находится нужный ему файл. Обращение к FTP-клиенту происходит по команде

ftp [<параметры>] [<имя сервера>] (5.1)

В качестве имени сервера указывается IP-имя или IP-адрес удаленного компьютера.

В большинстве серверов Internet для входа по FTP-команде нужны предварительная регистрация пользователя и указание пароля. Однако это не требуется при обращениях к общедоступным (анонимным) серверам. Такие серверы создают и обслуживают организации, заинтересованные в распростра-

нении информации определенного вида.

После выполнения команды (5.1) FTP-клиент переходит в командный режим. Примеры субкоманд, которые могут выполняться в командном режиме (ниже удаленный компьютер обозначен S, локальный компьютер — T):

open [<имя S>] — устанавливает связь с удаленным компьютером;

close [<имя S>] — разрывает связь с удаленным компьютером, оставаясь в командном режиме;

quit — то же, что и close, но с выходом из командного режима (из ftp);

cd [<имя каталога в S>] — выбор каталога на сервере;

get [<имя файла в S>[<имя файла в T >]] — перепись файла с S на T;

mget [<имена файлов в S>] — то же, что и get, но нескольких файлов;

put [<имя файла в T>[<имя файла в S>]] — обратная перепись;

mput <имена файлов в S> — то же, что и put, но более одного файла;

user <имя/пароль> — идентификация пользователя на сервере.

Каждый обмен порождает два процесса. Управляющий (командный) процесс инициирован в течение всего сеанса связи, а процесс передачи файла — только на время передачи. Протокольные порты сервера имеют номера 20 и 21, у клиента могут быть различные номера портов, в том числе несколько в одно и то же время. На каждый процесс обмена создаются свои копии FTP-сервера и клиента.

С помощью *протокола эмуляции терминала Telnet* пользователь сети Internet может работать на удаленном компьютере. Связь устанавливается при обращении к Telnet-программе командой

telnet <имя базы данных или системы каталогов> | <имя удаленного компьютера S>

После установления связи все данные, которые пользователь набирает на клавиатуре своего компьютера, передаются в S, а содержимое экрана S отображается на экране пользователя. Примерами команд в клиентской программе могут служить: установление связи (open), возвращение в командный режим клиентской программы Telnet (close), завершение работы (quit).

Telnet должен иметь возможность работать в условиях разных аппаратных платформ клиента и сервера. Это требование выполняется с помощью промежуточного виртуального терминала (аналогично SQL сервису в ODBC). В терминале зафиксирована интерпретация различных символов управления, поскольку их разновидностей не так уж много.

Телеконференции — доступ к информации, выделенной для группового использования.

Телеконференции могут быть глобальными или локальными. Включение материалов в телеконференцию, рассылка извещений о новых поступивших материалах, выполнение заказов — основные функции программного обеспечения телеконференций. Возможны режимы E-mail и on-line.

Самая крупная система телеконференций — USENET. В этой системе информация организована иерархически. Сообщения рассылаются или лавинообразно, или через списки рассылки. В режиме on-line можно прочитать список сообщений, а затем и выбранное сообщение. В режиме off-line из списка выбирается сообщение и на него посылается заказ.

Телеконференции могут быть с модератором (руководителем) или без него.

Существуют также средства аудиоконференций (голосовых телеконференций). Вызов, соединение, разговор происходят для пользователя как в обычном телефоне, но связь идет через Internet.

Электронная “доска объявлений” BBS (Bulletin Board System) — технология, близкая по функциональному назначению к телеконференции, позволяет централизованно и оперативно направлять сообщения для многих пользователей. Программное обеспечение BBS сочетает в себе средства электронной почты, телеконференций и обмена файлами.

В настоящее время интенсивно развиваются технологии настольной конференц-связи в реальном масштабе времени. Возможны несколько уровней настольной конференц-связи.

В зависимости от вида разделяемой пользователями информации различают уровни: простая E-mail сессия, совместная работа над проектом без голосовой связи (shared whiteboard — разделяемая “доска”), то же с голосовой связью (разновидность аудиоконференций), видеоконференция. По мере повышения уровня возрастают требования к пропускной способности используемых каналов переда-

чи данных. Для простых видов конференц-связи, а также и для аудиоконференций (конечно, при применении современных эффективных способов сжатия информации) можно использовать даже обычные телефонные линии, начиная с пропускной способности 8-10 кбит/с. Но лучше использовать в качестве “последней мили” цифровую ISDN или xDSL линию.

В зависимости от числа участников и способа интерактивной связи между ними различают двухточечную (unicast), широковещательную (broadcast) и многоточечную (multicast) телеконференции. Если в широковещательной конференции информация от центрального узла доставляется всем участникам, то в многоточечной конференции она рассылается избирательно, т.е. одновременно может идти обмен разной информацией внутри нескольких подгрупп одной группы пользователей.

Очевидно, что применение настольной конференц-связи в проектных организациях повышает эффективность работы, поскольку упрощает процесс принятия и согласования проектных решений, сокращает непроизводительные затраты времени на организацию совещаний, консультаций и т.п.

Программное обеспечение телеконференций включает в себя серверную и клиентскую части.

В клиентской программе должны быть, как минимум, средства E-mail, многооконный текстовый редактор (так, принимаемый и отправляемый партнеру тексты помещаются в разные окна, отдельное окно может быть выделено для видеоинформации в случае видеоконференций), средства файлового обмена. Наиболее известными клиентскими программами являются ProShare (Intel) и NetMeeting (Microsoft).

Серверная часть служит для распределения потока данных между пользователями с согласованием форматов окон с видеоинформацией, способов сжатия данных, скоростей потоков, идущих от разных сетей (пользователей). Примеры серверов: Whute Pine’s Meeting Point для видеоконференций, DataBeam’s Learning Server для систем дистанционного обучения.

Видеоконференция — способ связи, при котором осуществляется передача видеоизображений по телекоммуникационным каналам связи с возможностями интерактивного общения (в режиме on-line). Очевидно, что требования к пропускной способности каналов передачи данных в видеоконференциях существенно выше, чем в обычных телеконференциях. Видеоконференции стали доступными (для достаточно крупных организаций) после развития высокоскоростных каналов связи и эффективных алгоритмов сжатия данных при их передаче. В настоящее время широко внедряются сравнительно недорогие (от 1,5 до 7 тыс. долл.) настольные системы видеоконференц-связи.

Специализированная видеоконференц-система содержит дистанционно управляемую видеокамеру, монитор с большим экраном, микрофоны, динамики, устройство для считывания графических документов, кодеки – устройства для прямого и обратного преобразований информации из исходной в сжатую форму (кодек — совокупность первых слогов слов кодирование и декодирование). Цена комплекта — не менее 100 тыс. долл., что дешевле аналогового телевидения. Требуется выделенный канал со скоростью выше 64 кбит/с. Пример программного обеспечения — PictureTel.

В случае проведения видеоконференции для двух собеседников на базе ПЭВМ или рабочих станций (двухточечные настольные видеоконференции) требуется применение мультимедийных средств. Используются компьютер с аудио-, видео- и сетевой платами, микрофон, динамик, видеокамера. Примеры ПО: Intel Proshare или Sharevision, работающие под Windows. Эти системы можно использовать с телефонными линиями и высокоскоростными модемами, но качество будет низкое. Так, при 28,8 кбит/с частота кадров 7...10 Гц, размер окна 176×144 пикселей. При использовании ISDN можно повысить частоту кадров до 10...30 Гц. В большинстве систем предусмотрено наличие дополнительного окна, в котором виден совместно разрабатываемый документ.

Информационная система WWW (World Wide Web — всемирная паутина) — гипертекстовая информационная система сети Internet. Другое ее краткое название — Web.

Гипертекст — структурированный текст с введением в него перекрестных ссылок, отражающих смысловые связи частей текста. Слова-ссылки выделяются цветом и/или подчеркиванием. Выбор ссылки вызывает на экран связанный со словом-ссылкой текст или рисунок. Можно искать нужный материал по ключевым словам.

Информация, доступная в Internet по Web-технологии, хранится в Web-серверах (сайтах). Сервер имеет программу Listener, постоянно отслеживающую приход на определенный порт запросов от клиентов. Сервер удовлетворяет запросы, посылая клиенту содержимое запрошенных Web-страниц

или результаты выполненных процедур.

Клиентские программы WWW называют *браузерами* (brousers). Имеются текстовые (например, Lynx) и графические (наиболее известны Netscape Navigator и MS Explorer) браузеры. Фирма Sun Microsystems разработала браузер HotJava. В браузерах имеются команды листания, перехода к предыдущему или последующему документу, печати, перехода по гипертекстовой ссылке и т.п. Из браузеров доступны различные сервисы, например, FTP, E-mail. Для подготовки материалов к их включению в базу WWW разработаны специальный язык HTML (Hypertext Markup Language) и реализующие его программные редакторы, например Internet Assistant в составе редактора Word или SiteEdit, подготовка документов предусмотрена и в составе большинства браузеров.

Для связи Web-серверов и клиентов разработан протокол HTTP, работающий на базе TCP/IP. Web-сервер получает запрос от браузера, находит соответствующий запросу файл и передает его для просмотра в браузер.

Популярными серверами являются Apache Digital для ОС Unix, Netscape Enterprise Server и Microsoft Internet Information Server (IIS), которые могут работать как в Unix, так и в Windows NT, и Netware Web Server, предназначенный для работы в ОС Netware. Эти серверы поддерживают язык CGI, имеют встроенный HTML-редактор. Во многих серверах поддерживается стандарт шифрования SSL (Secure Sockets Layer) для защиты передаваемых по сети данных от несанкционированного доступа.

Опыт показывает, что для крупных серверов предпочтительнее платформа Unix, тогда как для серверов с малым числом транзакций лучше подходит ОС Windows NT.

На базе HTML создан язык виртуальной реальности VRML (Virtual Reality Modeling Language) — в нем дополнительно можно использовать 3*D* графику.

В новых ОС ожидается появление специальных средств поиска информации в серверах Internet. Пример такой технологии RDF (Resource Definition Format) — упорядочение метаинформации наподобие библиотечных каталогов (классификация по содержанию). В настоящее время для облегчения поиска применяют информационно-поисковые системы (ИПС), располагаемые на доступных пользователям Internet серверах. В этих системах собирается, индексируется и регистрируется информация о документах, имеющихся в обслуживаемой группе Web-серверов. Индексируются или все значащие слова, имеющиеся в документах, или только слова из заголовков. Пользователю предоставляется возможность обращаться к серверу с запросами на естественном языке, со сложными запросами, включающими логические связки. Примером таких ИПС может служить AltaVista. Для функционирования AltaVista выделено шесть компьютеров, самый мощный из них — 10-процессорная ЭВМ Alpha-8400, база данных имеет объем в 45 Гбайт.

Язык HTML — гипертекстовый язык для заполнения информационных Web-серверов. Он описывает структуру документа, вид которого на экране определяется браузером.

Описание на HTML — это текст в формате ASCII и последовательность включенных в него команд (управляющих кодов, называемых также *дескрипторами,* или *тегами*). Эти команды расставляются в нужных местах текста, они определяют шрифты, переносы, появление графических изображений, ссылки и т.п. В редакторах WWW вставка команд осуществляется нажатием соответствующих клавиш. Так, в Internet Assistant, входящем как дополнение в редактор MS Word, текст и команды набираются в едином процессе.

Собственно команды имеют форму <XXX>, где вместо XXX записывается имя команды.

Структура текста в WWW имеет вид:

```
<HTML><HEAD>
<TITLE> Заголовок текста </TITLE>
</HEAD>
<BODY>
Текст документа
</BODY>
</HTML>
```

В клиентской области окна при просмотре появляется только текст, помещенный между командами <BODY> и </BODY>. Заголовок между командами <TITLE> и </TITLE> выполняет только служебные функции.

Приведем примеры команд HTML. Команды форматирования текста (дескрипторы *компоновки*):

<P> — конец абзаца;

 — перевод строки;

<HR> — перевод строки с печатью горизонтальной линии, разделяющей части текста;

<LISTING> Текст </LISTING> — для представления листингов программ;

<BLOCKQUOTE> Текст </BLOCKQUOTE> — для выделения цитат.

Команды форматирования заголовков (дескрипторы *стиля*):

<H1> Текст </H1> — текст печатается наиболее крупным шрифтом, используется для заголовков верхнего уровня;

<H2> Текст </H2> — для следующего уровня и т.д. вплоть до команды <H6>;

<PRE> Текст </PRE> — указанный текст представлен заданным при его записи шрифтом.

Команды форматирования символов представлены парными символами B, I, U; текст между открывающей и закрывающей командами будет выделен соответственно полужирным шрифтом, курсивом, подчеркиванием.

Дескрипторы *списка* :

Команды форматирования списков <OL> и <UL> используются для выделения пунктов списков с нумерацией или с пометкой специальным символом (например, *) соответственно; каждый пункт в списке должен начинаться с команды <LI>. В словарях и глоссариях удобно применять команды <DL> — начало списка, <DT> — перед каждым новым термином словаря и <DD> — перед текстом определения каждого термина.

Дескрипторы *связи :*

В командах вставки графики и гипертекстовых ссылок используются адреса вставляемого или ссылочного материала, называемые URL (Uniform Resourse Locator). Ссылаться можно как на определенные места в том же документе, в котором поставлена ссылка, так и на другие файлы, находящиеся в любом месте сети. Перед простановкой внутренней ссылки, т.е. ссылки на некоторую позицию в данном файле, нужно разместить метку в этой позиции. Тогда URL есть указание этой метки, например, URL= #a35 есть ссылка на метку a35. URL может представлять собой имя файла в данном узле сети или IP-имя другого узла с указанием местоположения файла в этом узле и, возможно, также метки внутри этого файла.

Команда вставки графики

<IMG SRC=”URL”[ALIGN=TOP|MIDDLE|BOTTOM][ALT=”text”]>

ALIGN — параметр выравнивания, указывает место в окне для расположения рисунка; ALT — задает текст, который выводится на экран вместо рисунка в текстовых браузерах типа Lynx. Сами изображения должны быть в определенном формате (обычно это gif или jpeg).

Экран может быть разделен на несколько окон (областей, фреймов) с помощью парного тега <FRAMESET>. В каждом окне помещается содержимое файла (текст, изображение) указанием источника в теге <FRAME>, например

<FRAME SRC=имя файла>.

Команда гипертекстовой ссылки

<A HREF=”URL” >Текст </A>

Текст в окне будет выделен цветом или подчеркиванием. Можно ссылаться на определенное место в документе. Тогда

<A HREF=”URL#метка”> Текст </A>

Сама метка в документе имеет вид

<A NAME=”метка”> Текст </A>

Ссылки на фрагменты данного документа можно упростить

<A HREF=”#метка” >Текст </A>

Включение рисунка выполняется с помощью дескриптора

<IMG SRC = “fgr.gif”> или <A HREF = “http://www.abc.ru/de.htm”><IMGSRC = “fgr.gif”></A>

где fgr.gif и www.abc.ru/de.htm — конкретные имена, взятые для примера.

Расширение языка HTML — это язык XML (подмножество языка из стандарта SGML). Другое направление развития HTML — его динамическая версия DHTML.

SGML (Standard Generalized Markup Language — стандартный обобщенный язык разметки) определяет форму документов в виде последовательности объектов данных. Объектные данные могут храниться в различных файлах. Их включение в финальный документ происходит в форматах, задаваемых в специальном файле DTD (Document Type Definition). Шаблоны DTD упрощают хранение и поиск документов в базах данных.

XML (Extensible Markup Language) позволяет использовать в документах типы элементов, создаваемые для конкретных приложений, в нем также используются шаблоны DTD. Расширение заключается в возможности представления в одном XML-объекте информации разных типов (текст, графические данные, видео, звук). Для обмена документами на XML между Web-узлами разработан протокол ICE (Information and Content Exchange).

Наиболее известен среди *языков создания Web-приложений* язык Java — язык и технология программирования сетевых приложений, разработанный фирмой Sun Microsystems для систем распределенных вычислений.

Особенности языка Java: объектно-ориентированный, прототипом является C++, но более прост в использовании (так, например, убраны указатели); введены многопотоковость (например, оператор

синхронизации), дополнительная защита от вирусов.

Для пользователей важны также следующие особенности языка:

— аппаратная независимость (мобильность) за счет создания приложений в виде байт-кодов для некоторой виртуальной машины (рис. 5.2) — каждая платформа интерпретирует эти байт-коды; благодаря введению компиляции потеря эффективности, присущая интерпретации, здесь менее значительна;

— интеграция с браузерами;

— используемые программные объекты могут располагаться в разных узлах, интерпретатор находит их и загружает в компьютер пользователя.



Рис. 5.2. Компоненты программного обеспечения для языка Java

Другими словами, в узле-клиенте достаточно иметь лишь браузер, остальные программы и данные можно получить по сети. Однако при этом обостряется проблема информационной безопасности. В связи с этим загружаемым из сети программам (их называют *апплетами*) обычно запрещается обновлять и читать файлы, кроме тех, которые находятся на хосте самого аплета.

Java-аплеты доступны из HTML-документов (обращение к ним через тег <applet>), хотя могут использоваться и независимо от них.

CGI (Common Gateway Interface — *общий шлюзовой интерфейс*) — ПО связи HTML браузеров с другими прикладными программами и (или) текстами, находящимися на серверной стороне. Программа CGI — посредник между браузером и приложениями. Обычно программа CGI находится на сервере в специальном каталоге CGI_BIN, она является обработчиком запросов, идущих от браузера. Обращение к файлу из этого каталога означает запуск соответствующего обработчика. Если браузер обращается к документу не в HTML формате, то CGI преобразует форму документа в HTML и возвращает ее браузеру. Пример CGI-программы — WebDBC, организующей связь Web-сервера через ODBC-драйверы с нужными СУБД.

Наряду с интерфейсом CGI существуют и более частные интерфейсы, например, ISAPI (Internet Server Application Program Interface) фирмы Microsoft или NSAPI фирмы Netscape.

JavaScript — язык и интерпретатор этого языка для генерации и управления просмотром составных гипертекстовых документов. JavaScript более прост, чем Java, и тексты JavaScript исполняются быстрее, чем тексты Java или запросы к CGI, поскольку обработчики событий JavaScript реализованы в браузере, а не в сервере. Тексты на JavaScript записываются непосредственно в HTML документе с помощью специальных тегов и имеют вид

<SCRIPT LANGUAGE=”javascript”><!--...//--></SCRIPT> (5.2)

где <!-- ...//--> — текст в виде комментария. Браузеры, не имеющие JavaScript-обработчиков, просто игнорируют комментарий, а современные браузеры исполняют записанные в (5.2) вместо многоточия команды. В отличие от Java программы на JavaScript полностью интерпретируются в браузере.

Рассмотренные языки являются основой для создания программ межплатформенной распределенной среды. При этом в настоящее время создание крупных корпоративных приложений чаще опирается на применение CGI.

Информационная безопасность. Проблема информационной безопасности (ИБ) выходит за рамки сетевой ОС. Назначение систем ИБ сводится к защите от несанкционированных доступа и модификации информации, а также восстановлению информации после разрушений. Функции систем ИБ: аутентификация, разграничение доступа, защита на сетевом уровне.

Аутентификация чаще всего выполняется с помощью паролей. Разработан сервер Kerberos, предназначенный для аутентификации пользователя, выходящего в сеть с любого узла. Целесообразна периодическая смена паролей, доступ к файлам пароля должен быть только у администратора и т.п.

Разграничение доступа должно обеспечиваться на нескольких уровнях. Так, существует четырехуровневая модель. На внешнем уровне устанавливаются права доступа к корпоративной сети извне и выхода из нее. На сетевом, системном и прикладном уровнях регламентируются права доступа к се-

тевым информационным ресурсам, ресурсам ОС и пользовательским данным соответственно. Другая модель устанавливает уровни входа в систему, доступа к БД, доступа к приложениям. Права доступа часто представляются трехразрядным восьмеричным кодом *ABC*, в котором *A* — права владельца, *B* — членов группы, *C* — остальных пользователей, а три бита выражают право чтения, записи и исполнения соответственно.

Например, в САПР Euclid Quantum права доступа контролирует администратор системы, задавая список ACL (Access Control List). В ACL указываются имена, роли пользователей и их права доступа, которые выбираются среди следующих вариантов: просмотр, копирование, модификация, стирание данных, создание новых версий проектов, редактирование самого ACL, изменение статуса данных (варианты статуса — данные, доступные только конкретному разработчику, доступные членам рабочей группы, представленные на утверждение, уже утвержденные).

Между общедоступными и секретными объектами в сети (между общедоступными и частными сетями) можно установить специальное ПО, называемое *брандмауэром* (или firewall), которое либо запрещает выполнение определенных действий на сервере, либо фильтрует пакеты, разрешая проход только от оговоренных узлов.

Борьба с перехватом сообщений на сетевом уровне осуществляется методами криптографии. Криптография — это наука об обеспечении безопасности данных путем их шифрования.

Различают симметричную и асимметричную схемы шифрования.

В *симметричных схемах* (другое название – схемы с закрытым ключом) секретный ключ должен быть известен как отправителю, так и получателю. Ключ – это дополнение к правилу шифрования, представленное некоторым набором символов (например, двоичным кодом), управляющее преобразованием сообщения из исходного в зашифрованный вид. Например, ключ может быть операндом в действиях, выполняемых алгоритмом шифрования. Различают алгоритмы шифрования, основанные на перестановках, заменах символов исходного сообщения или на комбинациях этих действий. В частности, шифрование сообщения, выраженного двоичным кодом, может сводиться к поразрядной операции логического сложения кодов ключа и исходного текста.

Чем чаще обновляются ключи, чем они длиннее, тем труднее злоумышленнику их рассекретить. Поэтому очевидна полезность периодической смены ключей. Однако в симметричных схемах их обновление требует передачи вновь вводимого секретного ключа **К** участникам связи. Если эта передача осуществляется по каналу связи, то требуется шифрование **К** с помощью некоторого другого секретного ключа **С.**

В *асимметричных схемах* (схемах с открытым ключом) шифрование производится открытым ключом, а дешифрование — секретным ключом, известным только получателю. Возможность асимметричного шифрования вытекает из наличия так называемых односторонних функций $\mathbf{Y} = f(\mathbf{X})$, для которых обратное преобразование $\mathbf{X} = f^{-1}(\mathbf{Y})$ относится к трудным задачам, требующим полного перебора вариантов. Однако использование в обратном преобразовании ключа, который и является секретным, делает вычисление **X** сравнительно простой процедурой. Случайно подобрать секретный ключ злоумышленник не может, так как полный перебор при достаточной длине ключа за приемлемое время практически не осуществим.

В настоящее время все большее распространение получает комбинация симметричных и асимметричных схем. При этом сообщение кодируется закрытым ключом **К** по симметричной схеме, но сам ключ **К** для каждого сообщения новый и передается в закодированном по асимметричной схеме виде вместе с сообщением. Получатель декодирует сначала ключ **К** своим закрытым ключом **С**, а затем и все сообщение ключом **К**. Такая комбинация выгодна, во-первых, тем, что труднее взломать защиту, во-вторых, получатель быстрее дешифрирует сообщения, так как алгоритмы симметричного дешифрования заметно более экономичны.

Одним из применений шифрования является электронная подпись, предназначенная для удостоверения подлинности документа, пересылаемого по сети. Документ (чаще его аннотация) перед отправкой шифруется секретным ключом отправителя, а дешифрируется открытым ключом получателя.

5.2. Назначение и состав системных сред САПР

Системные среды автоматизированных систем. Системы автоматизированного проектирования относятся к числу наиболее сложных и наукоемких автоматизированных систем (АС). Наряду с выполнением собственно проектных процедур необходимо автоматизировать также управление проектированием, поскольку сам процесс проектирования становится все более сложным и зачастую приобретает распределенный характер. На крупных и средних предприятиях заметна тенденция к интеграции САПР с системами управления предприятием и документооборота. Для управления столь сложными интегрированными системами в их составе имеется специальное ПО — системная среда САПР или АС.

Первые системные среды САПР, называвшиеся мониторными подсистемами или Framework (FW), появились на рубеже 70...80-х г.г. В настоящее время основными функциями системных сред САПР являются управление данными, управление проектированием, интеграция ПО, реализация интерфейса с пользователем САПР, помощь в разработке и сопровождении ПО САПР.

Термин Framework применительно к системным средам САПР был введен в 1980 г. фирмой Cadence — одним из пионеров в создании системных сред САПР. Кроме Cadence, тематикой Frameworks для САПР электронной промышленности занималось несколько ведущих в этой области фирм (Mentor Graphics, IBM, DEC, Sun Microsystems и др.), создавших международную ассоциацию CFI (CAD Framework Initiative). Широкую известность получили такие системные среды, как Falcon Framework фирмы Mentor Graphics, Design Framework-2 фирмы Cadence и JCF (Jessy-Common Framework) европейской программы ESPRIT.

Важно отметить, что проблема системных сред САПР, зародившаяся в процессе становления САПР электронной промышленности, получила развитие при реализации CALS-технологии в различных отраслях машиностроения.

В типичной структуре ПО системных сред современных САПР можно выделить следующие подсистемы.

Ядро отвечает за взаимодействие компонентов системной среды, доступ к ресурсам ОС и сети, возможность работы в гетерогенной среде, настройку на конкретную САПР (конфигурирование) с помощью специальных языков расширения.

Подсистема управления проектом, называемая также подсистемой сквозного параллельного проектирования CAPE (Concurrent Art-to-Product Environment), выполняет функции слежения за состоянием проекта, координации и синхронизации параллельно выполняемых процедур разными исполнителями. Примерами подсистем управления проектами в машиностроении могут служить Design Manager в САПР Euclid, UG/Manager в Unigraphics. Иногда в отдельную подсистему выделяют управление методологией проектирования. При этом под *методологией* понимают совокупность методов и средств образования *маршрутов* проектирования — последовательностей проектных операций и процедур, ведущих к цели проектирования.

Методы построения маршрутов проектирования (workflow) зависят от типа проектных задач. Различают простые задачи, выполняемые одной программой, линейные, в которых нет разветвлений в межпрограммных связях, и комплексные. Методы построения маршрутов могут быть основаны на предварительном описании задач или на предварительном описании правил конструирования задач. В описании задач фигурируют порты, с которыми соотнесены данные. Порты могут быть обязательными и необязательными, порождающими дополнительные данные или данные нового объекта. Описания задач даются в виде графов или на языках расширения.

Подсистема управления методологией проектирования представлена в виде базы знаний. В этой базе содержатся такие сведения о предметной области, как информационная модель (например, в виде диаграмм сущность-отношение), иерархическая структура проектируемых объектов (например, в виде И-ИЛИ-дерева), описания типовых проектных процедур, типовые фрагменты маршрутов проектирования — так называемые потоки процедур, соответствие между процедурами и имеющимися пакетами прикладных программ, ограничения на их применение и т.п. Часто такую БЗ дополняют обучающей подсистемой, используемой для подготовки специалистов к использованию САПР.

Современные *системы управления проектными данными* называют PDM (Product Data Manager), иногда применительно к АСУ используют название EDM (Enterprise Data Manager). PDM предназначены для информационного обеспечения проектирования и выполняют следующие функции:

— хранение проектных данных и доступ к ним, в том числе ведение распределенных архивов документов, их поиск, редактирование, маршрутизация и визуализация;

— управление конфигурацией изделия, т.е. ведение версий проекта, управление внесением изменений;

— создание спецификаций;

— защита информации;

— интеграция данных (поддержка типовых форматов, конвертирование данных).

Основной компонент PDM — *банк данных* (БнД). Он состоит из системы управления базами данных и баз данных (БД). Межпрограммный интерфейс в значительной мере реализуется через информационный обмен с помощью банка данных. PDM отличает легкость доступа к иерархически организованным данным, обслуживание запросов, выдача ответов не только в текстовой, но и в графической форме, привязанной к конструкции изделия. Поскольку взаимодействие внутри группы проектировщиков в основном осуществляется через обмен данными, то в системе PDM часто совмещают функции управления данными и управления параллельным проектированием.

Подсистема интеграции ПО предназначена для организации взаимодействия программ в маршрутах проектирования. Она состоит из ядра, отвечающего за интерфейс на уровне подсистем, и оболочек процедур, согласующих конкретные программные модули, программы и/или программно-методические комплексы (ПМК) со средой проектирования.

Интеграция ПО базируется на идеях объектно-ориентированного программирования. Следует различать синтаксический и семантический аспекты интеграции. Синтаксическая интеграция реализуется с помощью унифицированных языков и форматов данных, технологий типа ODBC для доступа к общему банку данных или компонентно-ориентированных (CBD — Component-Based Development) технологий. Пример унифицированного формата — TES (Tool Encapsultion Specification), предложенного консорциумом CFI. Информация из TES используется для создания оболочек модулей при инкапсуляции. Семантическая интеграция подразумевает автоматическое распознавание разными системами смысла передаваемых между ними данных и достигается значительно труднее.

Подсистема пользовательского интерфейса включает в себя текстовый и графический редакторы и поддерживается системами многооконного интерфейса типа X Window System или Open Look.

Подсистема CASE предназначена для адаптации САПР к нуждам конкретных пользователей, разработки и сопровождения прикладного ПО. Ее можно рассматривать как специализированную САПР, в которой объектом проектирования являются новые версии подсистем САПР, в частности, версии, адаптированные к требованиям конкретного заказчика. Другими словами, такие CASE-подсистемы позволяют пользователям формировать сравнительно с малыми затратами усилий варианты прикладных ПМК из имеющегося базового набора модулей под заданный узкий диапазон конкретных условий проектирования. В таких случаях CASE-подсистемы называют *инструментальными средами*.

CASE-система, как система проектирования ПО, содержит компоненты для разработки структурных схем алгоритмов и “экранов” для взаимодействия с пользователем в интерактивных процедурах, средства для инфологического проектирования БД, отладки программ, документирования, сохранения “истории” проектирования и т.п. Наряду с этим, в CASE-подсистему САПР входят и компоненты с специфическими для САПР функциями.

Так, в состав САПР Microstation (фирма Bentley Systems) включена инструментальная среда Microstation Basic и язык MDL (Microstation Development Language) с соответствующей программной поддержкой. Язык MDL — С-подобный, с его помощью можно лаконично выразить обращения к проектным операциям и процедурам. В целом среда Microstation Basic близка по своим функциям к среде MS Visual Basic, в ней имеются генератор форм, редактор, конструктор диалога, отладчик.

САПР Спрут (российская фирма Sprut Technologies) вообще создана как инструментальная среда для разработки пользователем потоков задач конструкторского и технологического проектирования в машиностроении с последующим возможным оформлением потоков в виде пользовательских версий САПР. Сконструированный поток поддерживается компонентами системы, в число которых входят графические 2*D* и 3*D* подсистемы, СУБД, продукционная экспертная система, документатор, технологический процессор создания программ для станков с ЧПУ, постпроцессоры.

Наиболее известной CASE-системой в составе САПР в настоящее время является описываемая ниже система CAS.CADE фирмы MatraDatavision, с ее помощью фирма разработала очередную версию Euclid Quantum своей САПР Euclid.

Подходы к интеграции ПО в САПР. Для создания ПО САПР так же, как и других сложных автоматизированных информационных систем, определяющее значение имеют вопросы интеграции ПО. Теоретической базой для создания технологий интеграции ПО в САПР являются:

1) методология автоматизированного проектирования, в соответствии с которой осуществляются типизация проектных процедур и маршрутов проектирования в различных предметных областях, выявление типичных входных и выходных данных процедур, построение информационных моделей приложений и их обобщение, сравнительный анализ альтернативных методов и алгоритмов выполнения типовых процедур;

2) объектно-ориентированная методология, в соответствии с которой множества сущностей, фигурирующих в процессах проектирования, подразделяются на классы, в классах появляются свои процедуры и типы данных с отношениями наследования. Эти классы могут быть инвариантными и прикладными. Их обобщение и унификация приводят к появлению таких понятий и средств, как интегрированные ресурсы и прикладные протоколы, фигурирующие в стандартах STEP, или унифицированные программные компоненты типа графических ядер конструкторских САПР. Именно наличие типовых процедур и единообразное толкование атрибутов объектов в рамках конкретных протоколов позволяют разным программным системам “понимать” друг друга при взаимодействии.

Наряду с типовыми графическими ядрами, известны типовые ПМК имитационного моделирования, конструирования деталей и механизмов, технологической подготовки производства и др. Возможность использования типовых программ в составе программных комплексов обусловлена именно унификацией интерфейсов при обменах данными.

В некоторых маршрутах проектирования обмены данными должны происходить с высокой частотой, что обусловливает специфические требования к интерфейсам. Примером могут служить задачи имитационного моделирования, в которых требуется имитировать взаимодействие процессов, описываемых с помощью различного МО (например, на сосредоточенном и распределенном иерархических уровнях, или с помощью аналоговых и дискретных моделей). Для таких задач при моделировании характерно воспроизведение временной последовательности событий, происходящих в анализируемых взаимодействующих системах. Соответственно взаимодействие программ моделирования может происходить через фиксированное число временных шагов или по мере совершения тех или иных событий в моделируемых системах.

Так, в программах смешанного аналого-дискретного моделирования электронных устройств аналоговая часть моделируется с помощью программы анализа электронных схем, а дискретная часть — с помощью программы логического моделирования. Влияние аналоговой части на дискретную отображается в математических моделях путем преобразования непрерывных фазовых переменных в логические переменные в местах сопряжения частей модели, обратное влияние выражается в преобразовании идеализированных логических сигналов в заданные функции времени, соответствующие электрическим сигналам заданной формы. Очевидно, что в содержательной части сообщений, передаваемых из одной части в другую, должны быть сведения либо о состояниях, выражаемых значениями фазовых переменных в интерфейсных узлах, либо о событиях — изменениях фазовых переменных. Обмен сообщениями может происходить многократно в течение акта одновариантного анализа.

В программно-методических комплексах конструирования происходит обработка графической информации. Содержательная часть сообщений относится к геометрическим элементам, их размерам и положению в пространстве. В программах технологической подготовки механической обработки деталей наряду с геометрической информацией о конструкциях заготовок в передаваемые сообщения могут входить сведения об инструменте, технологической оснастке, оборудовании, режимах обработки, нормах времени, траекториях движения инструмента и рабочих органов оборудования и т.п.

Другими словами, в каждом приложении совокупность используемых при обменах понятий, предметных переменных и числовых параметров существенно ограничена и достаточно определена для того, чтобы можно было ставить вопрос о типизации моделей и языка взаимодействия. Такие вопросы решаются в рамках технологий STEP/CALS. Число приложений, нашедших свое описание в прикладных протоколах STEP ограничено, но совокупность таких протоколов может расширяться.

Прикладные протоколы STEP представляют семантическую сторону интеграционных технологий. Для интеграции нужна не только унификация моделей приложений, но и унификация механизмов взаимодействия, примерами которых являются технологии OLE, DDE, а также компонентно-ориентированные технологии.

Технологии интеграции ПО типа DDE и OLE. Современные ОС позволяют работать одновременно с несколькими задачами с выделением каждой задаче своего окна на экране дисплея. Межпрограммные взаимодействия осуществляются путем посылки сообщений, как это принято в объектно-ориентированном программировании. Используются специальные средства организации взаимодействий.

Так, ОС Unix поддерживает взаимодействие асинхронных параллельных процессов, в том числе в разных узлах сети. Каждый клиент должен предварительно зафиксировать свои потребности в виде имен используемых сообщений. Сообщения имеют структуру фрейма. Получатель сообщения определяет, что сообщение относится к нему, вызывает обработчик сообщения и использует полученные данные в соответствии с своими функциями.

В операционных системах Microsoft для организации межпрограммных взаимодействий были предложены средства Clipboard, DDE, OLE и в дальнейшем технология ActiveX.

Работа Clipboard основана на традиционном способе обменных зон — выделении кармана (некоторой области оперативной памяти, разделяемой взаимодействующими программами). При обменах одна программа посылает сообщение в карман, а другая извлекает, интерпретирует и использует это сообщение. Аналогичный режим работы осуществляется с помощью технологии формирования составных документов OLE, но здесь расширены возможности комбинирования данных различных типов в передаваемых документах.

Различают два способа взаимодействия: связь (linking) и внедрение (embedding). При связи в создаваемый документ включается не сам текст из источника, а лишь ссылка на него. Очевидно, что здесь меньше затраты памяти, изменения в источнике автоматически переходят в документ. При внедрении текст из источника физически переносится в документ. После чего документ можно редактировать независимо от источника. Оба этих способа реализованы в технологии OLE, что и зафиксировано в ее названии (OLE — Object Linking and Embedding).

При обмене с помощью DDE (Dynamic Data Exchange) программа-клиент запрашивает режим диалога с программой-сервером. В сообщении указывается имя сервера, имя раздела (обычно раздел — это файл), имя элемента (обмениваемая порция информации). Предварительно такой элемент (атом) должен быть создан, его адрес зафиксирован в таблице атомов. В ответ на запрос создается канал, по которому сервер передает данные или, что реализуется чаще, пересылает адрес нужного атома. По этому адресу клиент дополнительной командой может получить данные.

Подход к реализации интероперабельности, имеющийся в DDE и OLE, получил развитие в современных компонентно-ориентированных технологиях разработки ПО, рассматриваемых ниже.

Управление данными в САПР. В большинстве автоматизированных информационных систем применяют СУБД, поддерживающие реляционные модели данных.

Среди общих требований к СУБД можно отметить: 1) обеспечение целостности данных (их полноты и достоверности); 2) защита данных от несанкционированного доступа и от искажений из-за сбоев аппаратуры; 3) удобство пользовательского интерфейса; 4) в большинстве случаев важна возможность распределенной обработки в сетях ЭВМ.

Первые два требования обеспечиваются ограничением прав доступа, запрещением одновременного использования одних и тех же обрабатываемых данных (при возможности их модификации), введением контрольных точек (checkpoints) для защиты от сбоев и т.п.

Банк данных в САПР является важной обслуживающей подсистемой, он выполняет функции информационного обеспечения и имеет ряд особенностей. В нем хранятся как редко изменяемые данные (архивы, справочные данные, типовые проектные решения), так и сведения о текущем состоянии различных версий выполняемых проектов. Как правило, БнД работает в многопользовательском режиме, с его помощью осуществляется информационный интерфейс (взаимодействие) различных подсистем САПР. Построение БнД САПР — сложная задача, что обусловлено следующими особенностями САПР:

1. Разнообразие проектных данных, фигурирующих в процессах обмена как по своей семантике (многоаспектность), так и по формам представления. В частности, значительна доля графических данных.

2. Нередко обмены должны производиться с высокой частотой, что предъявляет жесткие требо-

вания к быстродействию средств обмена (полагают, что СУБД должна работать со скоростью обработки тысяч сущностей в секунду).

3. В САПР проблема целостности данных оказывается более трудной для решения, чем в большинстве других систем, поскольку проектирование является процессом взаимодействия многих проектировщиков, которые не только считывают данные, но и изменяют их, причем в значительной мере работают параллельно. Из этого факта вытекают следствия: во-первых, итерационный характер проектирования обычно приводит к наличию по каждой части проекта нескольких версий, любая из них может быть принята в дальнейшем в качестве основной, поэтому нужно хранить все версии с возможностью возврата к любой из них; во-вторых, нельзя допускать использования неутвержденных данных, поэтому проектировщики должны иметь свое рабочее пространство в памяти и работать в нем автономно, а моменты внесения изменений в общую БД должны быть согласованными и не порождать для других пользователей неопределенности данных.

4. Транзакции могут быть длительными и трудоемкими. *Транзакцией* называют последовательность операций по удовлетворению запроса. В САПР внесение изменений в некоторую часть проекта может вызвать довольно длинную и разветвленную сеть изменений в других его частях из-за существенной взаимозависимости компонентов проекта (многошаговость реализации запросов). В частности, транзакции могут включать в себя такие трудоемкие операции, как верификация проектного решения с помощью математического моделирования. В результате транзакции могут длиться даже несколько часов и более. Одна из трудностей заключается в отображении взаимозависимости (ассоциативности) данных. При хранении компонентов проекта во внешней памяти затраты времени на обработку запросов оказываются значительно выше, чем в большинстве других автоматизированных систем, с менее выраженными взаимозависимостями данных.

5. Иерархическая структура проектных данных и, следовательно, отражение наследования в целях сокращения объема базы данных.

В определенной мере названные особенности учитываются в СУБД третьего поколения, в которых стали применяться черты объектно-ориентированных (объектных) СУБД. В них наборы данных, характеризующих состояние предметной области (состояние проекта в случае САПР), помещаются в отдельные файлы. Интерпретация семантики данных осуществляется с помощью специальных процедур (методов), сопровождающих наборы. Наследование свойств объектов предметной области выражается с помощью введения категорий класса, надкласса, подкласса. Информационные модели приложений для таких СУБД разрабатываются на основе методик типа IDEF1X.

Объектные БД выгодны, во-первых, тем, что данные по конкретным объектам проектирования не разбросаны по множеству таблиц, как это имеет место в реляционных БД, а сосредоточены в определенных местах. Во-вторых, для каждого объекта могут быть назначены свои типы данных. В результате проще решаются задачи управления и удовлетворения запросов.

Наряду с чисто объектными СУБД (pure ODBMS), применяют СУБД объектно-реляционные. В последних происходит объединение свойств реляционных и объектно-ориентированных СУБД: объектно-ориентированная СУБД снабжается непроцедурным языком запросов или в реляционную СУБД вводятся наследование свойств и классы. Непроцедурность входного языка обеспечивается использованием языка SQL. Его операторы непосредственно включаются в программы на языке С. Возможно написание дополнительных программ, интерпретирующих SQL-запросы.

Отличительные особенности СУБД третьего поколения: расширенный набор возможных типов данных (это абстрактные типы, массивы, множества, записи, композиции разных типов, отображение величин с значениями разных типов), открытость (доступность из разных языков программирования, возможность обращения к прикладным системам из СУБД), непроцедурность языка (общепринятым становится язык запросов SQL), управление асинхронными параллельными процессами, состояние которых отражает БД. Последнее свойство позволяет говорить о тесной взаимосвязи СУБД и подсистемы управления проектами DesPM.

Названные особенности управления данными в САПР нашли свое выражение в современных подсистемах управления проектными данными PDM.

В PDM разнообразие типов проектных данных поддерживается их классификацией и соответст-

вующим выделением групп с характерными множествами атрибутов. Такими группами данных являются описания изделий с различных точек зрения (аспекты). Для большинства САПР машиностроения характерными аспектами являются свойства компонентов и сборок (эти сведения называют Bill of materials — BOM), модели и их документальное выражение (основными примерами могут служить чертежи, 3*D* модели визуализации, сеточные представления для конечно-элементого анализа, текстовые описания), структура изделий, отражающая взаимосвязи между компонентами и сборками и их описаниями в разных группах.

Вследствие большого объема проектных данных и наличия ряда версий проектов PDM должна обладать развитой системой поиска нужных данных по различным критериям.

Рассмотренные особенности банков данных в САПР позволяют квалифицировать их как системы Data Warehouse (DW), т.е. хранилища данных. Для хранилищ данных характерен ряд особенностей, совпадающих с названными выше особенностями банков данных САПР: 1) длительное хранение информации, отражающей историю разработок; 2) частота операций чтения данных выше частоты операций обновления данных; 3) использование единых форматов для однотипных данных, полученных из различных источников (например, от разных программно-методических комплексов). Эти особенности позволяют управлять конфигурацией проектов, что, в частности, означает хранение в САПР всех версий проекта и, возможно, данных по проектам предыдущих разработок, удовлетворение сложных запросов, для ответа на которые требуется извлечение и обработка данных из различных частей хранилища (так называемая многомерная обработка). Модели данных в DW отличаются от реляционных моделей (RM): в RM использованием нормальных форм стремятся максимально уменьшить избыточность данных, что приводит к увеличению числа таблиц, но уменьшенных размеров, однако многомерный поиск, требующийся в DW, в множестве таблиц затруднен. Поэтому в DW чаще используется модель данных “звезда”, в которой имеется общая таблица фактов (Fact Table) и каждому факту ставится в соответствие несколько таблиц с необходимыми атрибутами. Целостность данных в DW обеспечивается проверкой и трансформацией данных (data cleaning), вводимых из внешних источников, наличием дисциплины обновления данных, централизованным хранением основной базы, при этом достаточное быстродействие поддерживается передачей копий определенных частей базы в локальные базы, называемые киосками данных (Data Mart) и ориентированные на отдельные группы пользователей.

Примером СУБД, учитывающей требования, предъявляемые со стороны САПР, является система IMAN фирмы EDS Unigraphics. Это система управления объектно-ориентированными базами данных, ее можно также назвать системой интеграции данных. Она выполняет функции подсистемы PDM, которые являются функциями хранения данных, управления доступом к ним, контроля вносимых изменений, создания спецификаций изделий, интегрирования прикладных подсистем. Внутри IMAN используется реляционная модель данных, а на интерфейсном уровне — объектно-ориентированная информационная модель. Для синхронизации изменений предусматривается блокировка доступа пользователей, если с БД уже начал работу некоторый пользователь.

Другими известными примерами подсистем управления проектными данными могут служить системы Optegra (фирма Computervision), Euclid Design Manager (Matra Datavision), ProPDM в составе САПР Pro/Engineer (PTC), TechnoDOCS (Российская фирма “Весть”).

Ряд фирм разрабатывает системы PDM, которые можно использовать как самостоятельные продукты и как подсистемы в автоматизированных системах проектирования и управления.

Примером может служить система PartY (фирма Лоция Софт), в которой предусмотрены функции управления конфигурацией изделий, управления проектными данными и документооборотом, графический пользовательский интерфейс, реализация архитектуры клиент-сервер.

Варианты управления данными в сетях АС. При сетевой организации АС информационное обеспечение может быть реализовано по одному из следующих вариантов: 1) FS — файловый сервер; 2) RDA — доступ к удаленным данным; 3) DBS — сервер баз данных; 4) AS — сервер приложений. Варианты различаются распределением между разными узлами сети функций хранения данных, управления данными, обработки данных в приложениях и интерфейса с пользователем. На рис. 5.3 место среды передачи данных

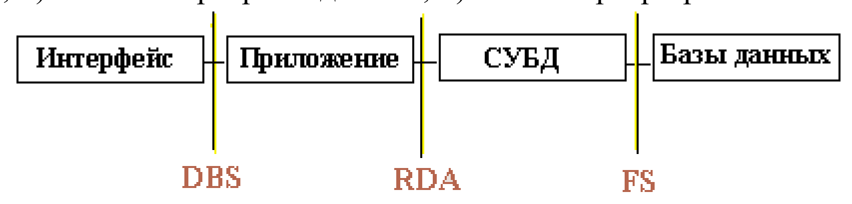


Рис 5.3. Варианты двухзвенных схем распределенных вычислений

показано вертикальной чертой для первых трех вариантов.

Каждый вариант имеет свою область применения.

Вариант файл-сервера характерен для локальных сетей на персональных ЭВМ с небольшим числом пользователей. Вследствие интенсивного трафика и трудностей с защитой информации эта структура для большинства АС малоэффективна. Поэтому предпочтительнее иметь СУБД в узле сервера. Вариант RDA — это модель удаленного узла, она наиболее распространена в настоящее время среди АС. В ней уменьшен трафик по сравнению с FS, унифицирован интерфейс с СУБД на основе языка SQL.

Примечание. Клиентов в FS и RDA иногда именуют "толстыми" клиентами, так как в них сосредоточены средства выполнения приложений.

Дальнейший переход к системе распределенных вычислений приводит к перемещению прикладного ПО или его части на специальный сервер или сервер БД, т.е. реализуются двух- и трехзвенные схемы. DBS — двухзвенная структура дистанционного управления, основанная на разделении прикладных процедур на две части: индивидуальные для каждого пользователя и общие для многих задач. В этой структуре под приложением понимают совокупность именно общих процедур. Эта совокупность обычно представляется на процедурных расширениях SQL и сохраняется в специальном словаре БД. В альтернативных вариантах (например, в RDA) все прикладные процедуры включаются в прикладные программы и, следовательно, при необходимости их изменения приходится модифицировать практически все прикладное ПО. Выделение таких процедур в отдельное приложение облегчает их модификацию. Кроме того, в DBS снижается трафик, так как обмены по сети происходят не для каждой операции с БД, а для каждой транзакции, состоящей из нескольких операций.

Вариант AS реализуется по *трехзвенной схеме*, в которой для приложений используются узлы, отделенные от терминального (локального) узла и от сервера БД, т.е. одновременно используются модели DBS и RDA.

Помимо проблемы распределения серверных функций между узлами сети, имеется проблема разделения этих функций между многими пользователями АС. Эта проблема решается либо по *схеме "один-к-одному"*, либо по *многопотоковой схеме*. В первой из них для каждого активного пользователя создается своя копия СУБД. Во второй СУБД должна быть реентерабельной программой, обслуживающей одновременно многих пользователей.

Интеллектуальные серверы БД. Особенности СУБД в таких сложных системах, как САПР, определяют их квалификацию, как *интеллектуальных* (их еще называют СУБД третьего поколения). К числу признаков интеллектуальной СУБД (дополнительно к охарактеризованным выше) относятся реализация в СУБД части прикладных процедур, что характерно для структуры DBS, *оповещение* пользователей (прикладных программ) об интересующих их изменениях состояния БД, *синхронизация событий* в БД, способность обслуживать прикладные программы, первоначально ориентированные на разные типы СУБД (назовем это свойство *многопротокольностью*).

Оповещение заключается в информировании программы *A* о совершении события, вызванного программой *B* и влияющего на работу программы *A* (рис. 5.4). Примером события может быть выход значения некоторого параметра в БД за допустимые пределы. Наиболее просто информирование можно организовать периодическим опросом состояния БД со стороны *A*. Однако это усложняет ПО и не эффективно по затратам времени и загрузке сети. Лучше возложить функцию оповещения на СУБД, что и присуще интеллектуальным СУБД. Но для этого нужно иметь обратные ссылки на программы, обращающиеся к БД, *правила* (иначе называемые *триггерами*), фиксирующие наступления событий, и процедуры обработки событий (см. рис. 5.4). Удобный вариант оповещения — информирование программы *A* о происшедших событиях во время ее активизации.

Для реализации многопротокольности разрабатывают специальные технологии. Наиболее известной среди них является технология ODBC (Open Data Base Connection) фирмы Microsoft. Фактически ODBC представляет собой библиотеку функций для обращений

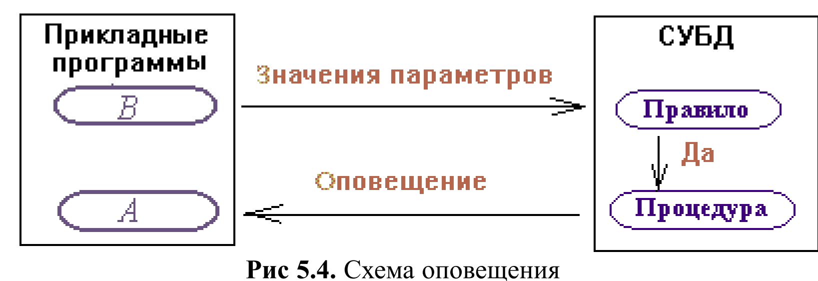


Рис 5.4. Схема оповещения

прикладных программ (ПП) к различным СУБД на основе языка SQL. Из ПП обращение происходит к виртуальной СУБД, в которой с помощью драйверов осуществляется переход к реальной СУБД.

Распределенные базы данных (РБД). В крупных АС, построенных на основе корпоративных сетей, не всегда удается организовать централизованное размещение всех баз данных и СУБД на одном узле сети. Поэтому появляются *распределенные базы данных*.

При построении РБД приходится решать ряд сложных проблем, связанных с минимизацией трафика, обеспечением интероперабельности обработки данных и целостности данных.

Минимизация трафика нужна в связи с тем, что при обслуживании запроса могут потребоваться данные из многих узлов, пересылаемые по сети. Возможности минимизации видны из примера обработки данных нескольких таблиц из разных узлов. Очевидно, что целесообразна однократная пересылка таблиц (причем таблиц именно меньшего размера) на один узел, на котором и будет обрабатываться запрос.

Интероперабельность выражает способность взаимодействия программ, работающих в гетерогенных сетях (в разных операционных средах или с разными СУБД). Интероперабельность обеспечивается или с помощью программ-шлюзов (конверторов) для каждой пары взаимодействующих сред, или с помощью единого унифицированного языка взаимодействия. Таким языком для доступа к БД является язык SQL, интероперабельность на его основе имеет место в системе ODBC (Open Data Base Connectivity), пример реализации которой показан на рис. 5.5. В примере СУБД FoxPro находится в локальном узле, а СУБД Ingres и Informix – в удаленных узлах. Прикладная программа имеет ODBC-интерфейс, не зависимый от особенностей различных СУБД. Менеджер драйверов реализует на базе унифицированного языка SQL все нюансы доступа к БД, общие для разных СУБД. Драйвер конкретной СУБД преобразует инвариантные к СУБД запросы в форму, принятую в данной СУБД. В трехзвенной структуре менеджер драйверов может быть размещен ан промежуточном сервере.

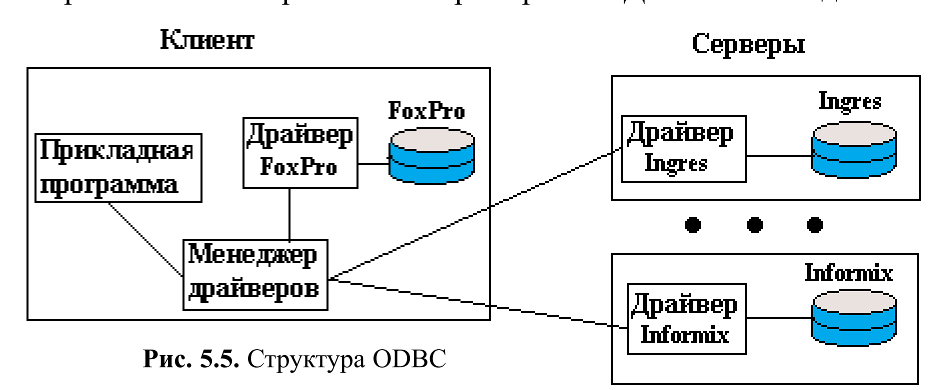


Рис. 5.5. Структура ODBC

Обеспечение целостности в РБД намного сложнее, чем в одноузловых БД. Различают два подхода к построению РБД: 1) тиражирование (репликация), при котором на нескольких серверах (узлах) сети расположены копии БД; 2) полномасштабная распределенность, при которой разные части БД находятся на разных серверах сети (классическая распределенность).

Применяют два способа тиражирования.

Способ, называемый репликацией первой копии, основан на выделении среди серверов с копиями БД одного первичного сервера (репликатора). Внесение изменений пользователями возможно только в БД первичного сервера, который в дальнейшем осуществляет тиражирование. *Тиражирование* — это перенос изменений БД из первичного сервера во все вторичные (локальные) серверы, которые используются клиентами только для чтения данных. Репликатор реагирует на события, фиксируемые триггерами, периодически пересылает обновленные данные в копии БД. Недостаток способа — невысокая надежность, присущая любым централизованным структурам.

Надежность повышается при использовании способа голосования. Здесь изменения посылаются не в один первичный, а в некоторые N серверов. При этом любой запрос на чтение направляется к некоторым M серверам, причем $N+M > K$, где K — общее число серверов. Принимается последняя по времени обновления версия ответа.

Тиражирование вносит избыточность в хранимые данные, появляются трудности с разрешением конфликтов из-за возможных несогласованных изменений в локальных БД. Однако по сравнению с классическими РБД, в которых данные не дублируются, заметно уменьшается трафик, надежнее и проще работа с локальными БД. Обеспечение надежности и удобства работы особенно актуально в случае ненадежных и медленных каналов связи, что имеет место во многих сетях в России.

В классических распределенных СУБД (РСУБД) необходимо *управлять одновременным досту-*

пом, что должно гарантировать целостность (сериализуемость) БД. Наиболее широко используются алгоритмы управления, основанные на механизме блокировки. При этом *блокировкой* называют ситуацию, при которой некоторая транзакция объявила о желании получить полномочия на доступ к странице памяти и, следовательно, другие транзакции не имеют права занимать этот ресурс.

Одним из способов управления является централизованное блокирование, при котором на одном из узлов поддерживается единая таблица блокировок. Такой узел устанавливает очередность выполнения транзакций, что исключает конфликты. Однако при централизованном управлении невысока надежность и требуется мощный сервер.

В РСУБД с репликацией нет проблемы согласования при записи действий многих узлов. Собственно тиражирование чаще всего выполняется по правилу полной эквивалентности — обновленные данные сразу же после изменившей их транзакции рассылаются по всем локальным БД. Чтение же выполняется из БД одного конкретного узла, наиболее близкого к пользователю в функциональном или географическом смысле.

Сложнее решать проблемы распределенного управления, что требуется в РСУБД без тиражирования. Одним из распространенных протоколов распределенного управления является протокол двухфазной фиксации транзакций (2PC). На первой фазе инициатор транзакции (координатор) рассылает участникам выполнения транзакции оповещения о блокировке. В ответ узлы сообщают о своей готовности или неготовности. На второй фазе координатор сообщает либо о “глобальной фиксации”, т.е. о выполнении транзакции, либо об откате транзакции. Неприятности возможны при сбоях, которые могут оставить некоторый узел в заблокированном состоянии: он не может ни выполнять транзакцию, ни отменять ее в одностороннем порядке.

Программные средства управления проектированием в САПР. В зависимости от степени автоматизации управляющих функций можно выделить несколько уровней управления проектированием:

1) компонентный; на этом уровне пользователь должен знать специфические особенности каждой конкретной программы, используемой в маршруте проектирования; при организации маршрута он должен позаботиться об информационных интерфейсах используемых программ; другими словами, системная среда лишь предоставляет сведения о имеющихся программах и их интерфейсах;

2) ресурсный; пользователь по-прежнему оперирует программами при компиляции маршрута проектирования, но системная среда позволяет скрыть специфику каждой программы, так как общение унифицировано;

3) задачный; пользователь составляет маршрут проектирования не из отдельных программ, а из отдельных проектных процедур; покрытие маршрута программами выполняет системная среда;

4) проблемный; пользователь формулирует задания в форме “что нужно сделать”, а не “как это сделать”, т.е. не определяет маршрут проектирования, а ставит проектную проблему.

В системных средах САПР *управление проектированием* возлагается на подсистему CAPE, в некоторых системах обозначаемую как DesPM (Design Process Manager). DesPM должна включать в себя компоненты: комплексы базовых знаний по тем предметным областям, которые определяются объектом проектирования, а также знаний о языках представления характеристик и ограничений; средства для генерации плана (маршрута проектирования), определения наличия средств и ресурсов для реализации плана; средства выполнения плана; средства оценки результатов. DesPM позволяет выбирать объекты проектирования, производить декомпозицию моделей, для каждого компонента выбирать проектные процедуры из имеющегося набора.

По каждому объекту DesPM выдает сообщения, примерами которых могут быть: “объект проектируется другим разработчиком”, “проектирование преждевременно, не выполнены предшествующие процедуры”, “не подготовлены исходные данные”. Одной из важнейших функций DesPM является помощь в реализации параллельного проектирования. Желательно в DesPM предусмотреть возможности создания “суперпроцедур” — командных файлов для выполнения часто повторяющихся фрагментов маршрутов проектирования.

Расширение возможностей управления проектированием и адаптация системной среды к конкретным САПР связано с применением языков расширения. *Язык расширения* — это язык программирования, позволяющий адаптировать и настраивать системную среду САПР на выполнение новых

проектов. Язык расширения должен обеспечивать доступ к различным компонентам системной среды, объединять возможности базового языка программирования и командного языка, включать средства процедурного программирования. Для большинства языков расширения базовыми являются ЛИСП или С.

Так, язык Skill из Design Framework-2 фирмы Cadence или язык CCL (CASE Comment Language) фирмы Matra Datavision являются ЛИСП-подобными, а язык AMPLE из Falcon Framework фирмы Mentor Graphics базируется на языках *С* и ПАСКАЛЬ.

Управление процессом проектирования включает в себя большое число действий и условий, поддерживающих параллельную работу многих пользователей над общим проектом. Управление выполняется на основе моделей вычислительных процессов. Используются спецификации моделей, принятые в CASE-системах, например, диаграммы потоков данных, ориентированные графы. Сначала модели составляют для задачного уровня, а затем система осуществляет их покрытие. Применяют также описания на языках расширения или 4GL. В системной среде Ulyses спецификации даны в виде набора модулей с указанием условий их активизации, что близко к представлению моделей в системах, управляемых знаниями. Так, каждый проектирующий программный модуль может быть активизирован только в том случае, если входные данные готовы. Для этого специальная *программа управления модулями* системной среды отслеживает соблюдение отношений следования между проектными операциями и процедурами, заданными в маршруте проектирования. На эту же программу возлагаются функции регулирования прав доступа к модулям, сбор статистики (протоколирование) по обращениям к модулям и некоторые другие.

Необходимо обеспечение синхронизации изменения данных, разделяемых многими пользователями. Для этого, во-первых, пользователи подразделяются на классы (администрация системы, руководство проектом и частями проекта, группы исполнителей-проектировщиков) и для каждого класса вводят определенные ограничения, связанные с доступом к разделяемым данным; во-вторых, обеспечивают средства ведения многих версий проекта; в-третьих, для выполнения работ в отдельных ветвях параллельного процесса пользователям выделяют свои рабочие области памяти. Данным могут присваиваться различные значения статуса, например, “правильно”, “необходимо перевычисление”, “утверждено в качестве окончательного решения” и т.п. Собственно синхронизация выполняется с помощью механизмов типа рандеву или семафоров, рассматриваемых в пособиях по параллельным вычислениям.

Примером подсистемы управления проектированием в САПР СБИС может служить Minerva, разработанная специалистами университета Карнеги-Меллона (США). В ней реализуется нисходящее проектирование на основе модели в виде И-ИЛИ-дерева. Дерево может быть не полностью определено к началу проектирования и его отдельные кусты дорабатываются в процессе проектирования. На каждом ярусе дерева происходит выбор альтернатив, формирование ТЗ для следующего иерархического уровня, возможны возвраты. В средствах пользовательского интерфейса предусмотрено высвечивание на экране фрагментов дерева, по каждой ветви дерева сообщается о ее готовности к проработке, занимается ли ею кто-то другой из разработчиков и т.п.

В общем случае полная формализация управления проектированием не может быть достигнута, поэтому полезную роль играют *системы поддержки решений*, принимаемых людьми, DSS (Decision Support Systems). В качестве таких систем часто используют хранилища данных и OLAP-средства (On-Line Analytical Processing).

Использование хранилищ данных имеет ряд преимуществ в управлении большими объемами данных: имеется единое ядро, что исключает чрезмерно разветвленные и длительные транзакции, легче синхронизировать внесение изменений, поддерживать единство форматов данных, хранить предыдущие версии и т.п.

OLAP-средства должны обеспечивать оперативный доступ к данным, на основе которого выявляются зависимости между параметрами (измерениями в многомерной модели приложения). В OLAP-системах на реляционных СУБД аналитическая обработка, или, другими словами, многомерный динамический анализ данных требует просмотра большого числа записей из разных таблиц. Поэтому производительность оказывается невысокой. В специализированных OLAP-системах, обеспечивающих более быстрый многомерный анализ, но с более существенными ограничениями на объем БД, данные хранятся в виде гиперкубов или поликубов — многомерных таблиц с постоянным или пе-

ременным числом ячеек соответственно. Пример OLAP системы — Oracle Express, помогающей менеджерам и аналитикам получать данные в виде разрезов таких многомерных таблиц, готовить отчеты, обосновывать решения.

В составе подсистем управления методологией проектирования полезно иметь средства консультирования по принятию проектных решений. Они могут быть представлены в виде множества модулей, объединяемых гипертекстовой оболочкой. Каждый модуль содержит некоторый совет по выбору решения, преодолению противоречий, возникающих в процессе проектирования. Здесь уместно использование методов и приемов решения изобретательских задач.

Примером программы консультирования и прогнозирования результатов принимаемых решений может служить программа Clio в упомянутой выше подсистеме Minerva.

Примеры подсистем управления данными и проектированием. В ряде системных сред САПР (прежде всего САПР в машиностроении) в подсистемах PDM объединяются функции управления данными и проектированием. Пример такой PDM — подсистема Design Manager в САПР Euclid Quantum. Функциями этой PDM являются управление потоками проектных данных, версиями проекта, взаимодействием разработчиков, защита информации, конфигурирование и адаптация версий системы для конкретных пользователей.

Подсистема Design Manager в Euclid Quantum состоит из частей пользовательской, администратора и управления структурой продукта.

В пользовательской части данные при выполнении проектирования могут находиться либо в распоряжении конкретного разработчика, в частности, в его индивидуальной БД (User Area), либо в зоне работы рабочей группы (Workgroup Area), в частности, в ее БД. Утвержденные данные пересылаются в центральную БД (Repository). Пересылка данных из User Area (*UA*) в Workgroup Area (*WGA*) происходит по инициативе разработчика командами *check in* или *share*. Первая из них начинает процедуру контроля данных, вторая обеспечивает разделение данных всеми участниками рабочей группы. Контроль данных выполняет уполномоченный член группы, результатом является или утверждение и, следовательно, направление их в репозиторий *R*, или неутверждение и отправка данных в *UA* на доработку. Разработчик может запрашивать данные для начала нового проекта по команде *copy out* или для модификации существующего проекта по команде *check out* (рис 5.6).

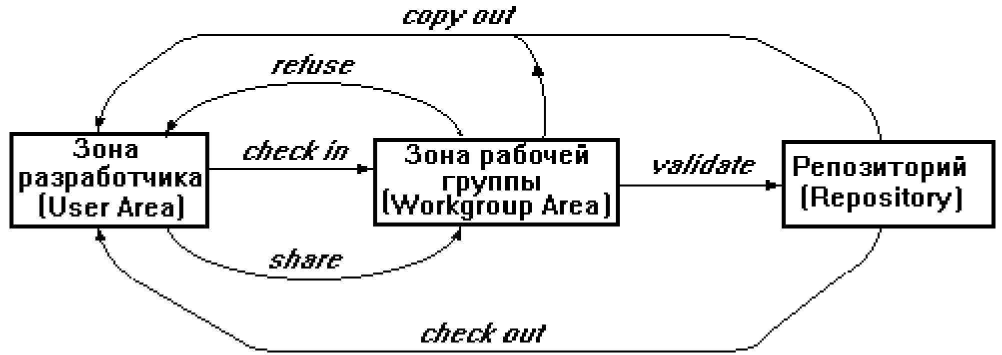


Рис.5.6. Потоки данных в PDM Design Manager (САПР Euclid Quantum)

В БД данные организованы иерархически, группируются по именам проектов или по типам данных. Вызов данных из любой БД (*UA, WGA*, *R*) выполняется командой *retrieve*, посылка в БД — командой *store*. При обращении к БД пользователь видит структуру данных (директорию — имена папок и их частей) и определенный аспект данных выделенного в директории проекта. Такими аспектами могут быть свойства документа (имя, автор, дата, статус и т.п.), список версий проекта, 3*D* изображение.

В функции администратора системы входят упорядочение данных с их распределением по дискам, контроль за правами доступа пользователей, связь с внешними системами (управление импортом/экспортом данных) и др.

В системной среде NELSIS CAD Framework имеются части: 1) DMS (Design Management Services) для поддержки иерархии данных, управления версиями и потоками задач; 2) DMI (Design Management Interface) с функциями открытия и закрытия баз данных, вызова и пересылки данных, доступа к DMS; 3) FUS (Framework User Services), включающая ряд браузеров для визуализации информации.

Базовая сущность в NELSIS CAD Framework – объект (ячейка). Объект состоит из нескольких примитивов и/или ссылок. Объекты объединяются в модули. В модуле все объекты имеют одни и те же имена и тип представления (viewtype) и являются вариантами описания одного и того же физического объекта, т.е. это версии или улучшения предыдущих вариантов. Объекты могут находиться в отношениях эквивалентности друг с другом или иерархии. Каждый модуль имеет атрибут, обозначающий уровень абстракции. Версии нумеруются и им присваивается тот или иной статус.

Предусмотрены следующие статусы: 1) *рабочий* – объект находится в работе, его можно модифицировать, в модуле хотя бы один объект должен иметь этот статус; в процессе модификаций новая версия может замещать старую или старая версия сохраняется, получая, например, статус Backup; 2) принятый (*actual* version) – именно эта версия служит для обмена между объектами, автоматически не стирается, ее модификации осуществляются через рабочий статус; 3) архивный (*Backup*); 4) порождаемый (*Derived* version) – статус зарезервирован для вновь создаваемых объектов, например, при синтезе проектных решений. Разработчик сам изменяет статус объектов.

Любое изменение должно отражаться в отношениях объекта. NELSIS CAD Framework не изменяет существующие отношения, а создает новые. Например, если изменяется объект “топология”, то новая версия не наследует отношение со схемой, которая была получена экстракцией из старой топологии. Целостность данных поддерживается тем, что нельзя одновременно работать и изменять один и тот же объект разным разработчикам, так как каждый из них будет работать со своей рабочей версией.

Данные проекта могут находиться в нескольких БД распределенного банка данных. Данные одной части проекта доступны другим частям, что позволяет выполнять параллельное проектирование.

Для интеграции программных компонентов в системную среду (т.е. для согласования по данным этих компонентов с БД среды) используются обычные модификации компонента, если известен его код, или создается оболочка – модульная абстракция.

В NELSIS CAD Framework имеется несколько браузеров для общения с пользователем. Для каждого браузера может быть открыто свое окно.

1. Design flow browser – показывает взаимосвязь между проектными процедурами, историю получения объекта, список процедур, которые могут быть выполнены над объектом, позволяет задавать маршруты проектирования, вызывать проектные процедуры и задавать их параметры,

2. Hierarchy Browser – показывает граф иерархии и место объекта в ней.

3. Version Browser – показывает все виды (viewtypes), статусы и номера версий выбранного объекта. Он может показать отношения эквивалентности, т.е. объекты, выражающие разные аспекты, например, топологию, схему, результаты моделирования физического объекта.

4. Equivalence Browser показывает отношения эквивалентности для выбранного объекта.

5. Schema Browser показывает сущности и их отношения в виде схемы данных, в отдельном окне показываются запросы к БД и ответы на них.

5.3. Инструментальные среды разработки программного обеспечения

Среды быстрой разработки приложений. CASE-системы часто отождествляют с инструментальными средами разработки ПО, называемыми *средами быстрой разработки приложений* (RAD — Rapid Application Development). Примерами широко известных инструментальных сред RAD являются VB (Visual Basic), Delphi, PowerBuilder фирм Microsoft, Borland, PowerSoft соответственно. Применение инструментальных сред существенно сокращает объем ручной работы программистов, особенно при проектировании интерактивных частей программ.

Большое практическое значение имеют инструментальные среды для разработки ПП, предназначенных для работы под управлением операционных систем Windows, в связи с широкой распространенностью последних.

Простейшая система для написания Windows-программ на языке C++, позволяющая сократить объем кода, создаваемого пользователем вручную, основана на библиотеке DLL (Dynamic Link Library), которая содержит модули, реализующие функции API (Application Programming Interface) для связи прикладных программ с ОС Windows.

Эта система получила развитие в MFC (Microsoft Foundation Classes), представляющей собой библиотеку классов для автоматического создания каркасов ПО многоуровневых приложений. В библиотеке имеются средства для поддержки оконного интерфейса, работы с файлами и др.

В средах быстрой разработки приложений RAD обычно реализуется способ программирования, называемый *управлением событиями*. При этом достигается автоматическое создание каркасов программ, существенно сокращается объем ручного кодирования. В этих средах пользователь может работать одновременно с несколькими экранами (окнами). Типичными являются окна из следующего списка.

1. Окно меню с пунктами “file”, “edit”, “window” и т.п., реализующими функции, очевидные из названия пунктов.

2. Окно формы, на котором собственно и создается прототип экрана будущей прикладной программы.

3. Палитра инструментов — набор изображений объектов пользовательского интерфейса, из ко-

торых можно компоновать содержимое окна формы.

4. Окно свойств и событий, с помощью которого ставятся в соответствие друг другу объекты окна формы, события и обработчики событий. *Событием* в прикладной программе является нажатие клавиши или установка курсора мыши в объект формы. Каждому событию должна соответствовать событийная процедура (обработчик события), которая проверяет код клавиши и вызывает нужную реакцию. В RAD имеются средства для удобства разработки обработчиков событий.

5. Окно редактора кода, в котором пользователь записывает создаваемую вручную часть кода.

6. Окно проекта — список модулей и форм в создаваемой программе.

Для написания событийных процедур в Visual Basic используется язык и текстовый редактор одноименного языка, в Delphi — язык и редактор языка Object Pascal. В CASE-системе фирмы IBM, включающей части VisualAge (для клиентских приложений) и VisualGen (для серверных приложений), базовым языком выбран SmallTalk. В среде разработки приложений клиент-сервер SQLWindows оригинальные фрагменты программ пишутся на специальном языке SAL. Нужно заметить, что для реализации вычислительных процедур и, в частности, для написания миниспецификаций используется обычная для 3GL технология программирования.

Обычно после написания ПП на базовом языке компилятор системы переводит программу на промежуточный *p*-код. Вместе с интерпретатором *p*-кода эта программа рассматривается, как EXE-файл. В некоторых развитых средах компилируется обычный EXE-файл, не требующий интерпретации для своего исполнения.

Помимо упрощения написания пользовательского интерфейса, в средах RAD предусматриваются средства для реализации и ряда других функций. Так, в наиболее развитой версии Visual Basic к ним относятся средства выполнения следующих функций:

— поддержка ODBC, что дает возможность работы с различными СУБД;

— разработка баз данных;

— разработка трехзвенных систем распределенных вычислений;.

— интерактивная отладка процедур на SQL Server;

— управление версиями при групповой разработке ПО;

— моделирование и анализ сценариев распределенных вычислений.

Для создания сред RAD в случае *сетевого программирования* требуется решить ряд дополнительных проблем, обусловливаемых многоплатформенностью в гетерогенных сетях, обилием применяемых форматов данных, необходимостью защиты информации и т.п. Решение этих проблем достигнуто в объектно-ориентированных технологиях на базе языка сетевого программирования Java. Кроме того, с помощью Java удается решить еще одну актуальную для Internet и Intranet задачу — сделать Web-страницы интерактивными.

Платформенная инвариантность в Java достигается, благодаря введению виртуальной метамашины с системой команд, максимально приближенной к особенностям большинства машинных языков. Любой Web-сервер при наличии запроса на Java-программу со стороны клиента транслирует (компилирует) эту программу на язык метамашины. Скомпилированный модуль, называемый байт-кодом, пересылается клиенту. Клиент должен выполнить интерпретацию байт-кода. Соответствующие интерпретаторы в настоящее время имеются в браузерах всех основных разработчиков Web-технологий.

Java используется двояким образом. Во-первых, как средство “оживления” Web-страниц. В этом случае программный Java-компонент называют *аплетом*, аплет встраивается в страницу с помощью специального тега, имеющегося в языке HTML. Во-вторых, Java — универсальный язык программирования и может быть использован для написания любых приложений, не обязательно привязанных к Web-технологии.

Хотя и ранее были известны технологии на базе промежуточных *p*-кодов, именно технология Java, оказалась наилучшим образом приспособленной для использования в гетерогенной сетевой среде. Она последовательно отражает принципы объектно-ориентированного программирования и обеспечивает приемлемую эффективность (производительность) исполнения программ. Эту эффективность можно еще более повысить, если в браузерах заменить интерпретацию на компиляцию.

Для разработки ПО на языке Java создан ряд инструментальных средств. Основной средой явля-

ется JDK (Java Developer's Kit). В ней имеются: 1) библиотеки классов, в том числе библиотеки основных элементов языка, часто используемых оболочек (wrapper), процедур ввода-вывода, компонентов оконного интерфейса и др. 2) инструментальные средства такие, как компилятор байт-кодов, интерпретатор, просмотрщик аплетов, отладчик, формирователь оконных форм и т.п. Развитую RAD-среду — PowerJ — предлагает фирма Sybase.

Значительное внимание уделяется разработке инструментальных сред для создания Web-узлов, примером такой среды может служить HAHTSite фирмы HAHTSoftware. Для разработки Java-программ из готовых компонентов можно использовать среду IBM Visual Age for Java, в которой имеются (как и в среде VB) версии учебная, профессиональная и общецелевая (Enterprise), и др.

Наряду с самостоятельными RAD-системами имеются и RAD-системы в составе САПР. Это прежде всего упомянутая выше система CAS.CADE фирмы Matra Datavision.

Компонентно-ориентированные технологии. Появление компонентно-ориентированных технологий вызвано необходимостью повышения эффективности разработки сложных программных систем, являющихся в условиях использования корпоративных и глобальных вычислительных сетей распределенными системами. Компонентно-ориентированные технологии основаны на использовании предварительно разработанных готовых программных компонентов.

Компиляция программ из готовых компонентов — идея не новая. Уже первые шаги в области автоматизации программирования были связаны с созданием библиотек подпрограмм. Конечно, для объединения этих подпрограмм в конкретные прикладные программы требовалась ручная разработка значительной части программного кода на языках третьего поколения. Упрощение и ускорение разработки прикладного ПО достигается с помощью языков четвертого поколения (4GL), но имеющиеся системы на их основе являются специализированными и не претендуют на взаимодействие друг с другом.

Современные системы интеграции ПО построены на базе объектной методологии.

Так, выше приведены примеры библиотек классов, применяя которые прикладные программисты могут создавать субклассы в соответствии с возможностями наследования, заложенными в используемые объектно-ориентированные языки программирования. При этом интероперабельность компонентов в сетевых технологиях достигается с помощью механизмов, подобных удаленному вызову процедур RPC. К библиотекам классов относятся MFC, библиотеки для доступа к реляционным БД (например, для встраивания в прикладную программу драйверов ODBC) и др.

Преимущества использования готовых компонентов обусловлены тщательной отработкой многократно используемых компонентов, их соответствием стандартам, использованием лучших из известных методов и алгоритмов.

В то же время в компонентах библиотек классов спецификации интерфейсов не отделены от собственно кода, следовательно, использование библиотек классов не профессиональными программистами проблематично. Именно стремление устранить этот недостаток привело к появлению CBD — компонентно-ориентированных технологий разработки ПО. Составными частями таких технологий являются унифицированные способы интеграции программного обеспечения.

Возможны два способа включения компонентов (модулей) в прикладную программу — модернизация (reenginering) или инкапсуляция (encapsulation или wrapping).

Модернизация требует знания содержимого компонента, интероперабельность достигается внесением изменений собственно в сам модуль. Такой способ можно назвать способом "белого ящика". Очевидно, что модернизация не может выполняться полностью автоматически, требуется участие профессионального программиста.

Инкапсуляция выполняется включением модуля в среду с помощью интерфейса — его внешнего окружения (оболочки — wrapper). При этом компонент рассматривается как "черный ящик": спецификации, определяющие интерфейс, выделены из модуля, а детали внутреннего содержимого скрыты от пользователя. Обычно компоненты поставляются в готовом для использования виде скомпилированного двоичного кода. Обращения к модулю возможны только через его интерфейс. В спецификации интерфейса включаются необходимые для интероперабельности сведения о характеристиках модуля — *модульная абстракция*. В состав этих сведений могут входить описания всех входных и выходных для модуля данных (в том числе имеющихся в модуле интерактивных команд), структура командной строки для инициализации процедур, сведения о требуемых ресурсах.

Компонентно-ориентированные системы построены на основе инкапсуляции компонентов. В архитектуре этих систем можно выделить следующие части: 1) прикладная программа (клиент), создаваемая для удовлетворения возникшей текущей потребности; 2) посредник (брокер или менеджер), служащий для установления связи между взаимодействующими компонентами и для согласования их интерфейсных данных; 3) множество компонентов, состоящих каждый из программного модуля, реализующего некоторую полезную функцию, и оболочки (интерфейса). В спецификации интерфейса могут быть указаны характеристики модуля, реализуемые методы и связанные с модулем события (например, реакции на нажатие клавиш).

Собственно интерфейс представляет собой обращения к функциям модуля, называемым в CBD-технологиях методами. Эти обращения переводятся в двоичный код, что обеспечивает при их использовании независимость от языка программирования. Один и тот же модуль может реализовывать несколько разных функций, поэтому у него может быть несколько интерфейсов или методов. Каждый новый создаваемый интерфейс обеспечивает доступ к новой функции и не отменяет прежние возможно еще используемые интерфейсы.

Схематично взаимодействие компонентов можно представить следующим образом. Клиент обращается с запросом на выполнение некоторой процедуры. Запрос направляется к посреднику. В посреднике имеется предварительно сформированный каталог (реестр или_репозиторий) интерфейсов процедур с указанием компонентов-исполнителей. Посредник перенаправляет запрос соответствующему исполнителю. Исполнитель может запросить параметры процедуры. После выполнения процедуры полученные результаты возвращаются клиенту.

При этом пользователь оперирует удобными для его восприятия идентификаторами компонентов и интерфейсов, а с помощью каталога эти идентификаторы переводятся в указатели (ссылки), используемые аппаратно-программными средствами и которые однозначно определяют интерфейс в распределенной сети из многих компьютеров.

В большинстве случаев реализуется синхронный режим работы, подразумевающий приостановку процесса клиента после выдачи запроса до получения ответа.

Наиболее популярными в настоящее время являются следующие CBD-технологии.

— OpenDoc — технология, основанная на спецификациях CORBA, разработанных в начале 90-х г.г. специально созданным консорциумом OMG, в который вошли представители ведущих компьютерных фирм. В OpenDoc реализуется технология распределенных вычислений на базе программ-посредников ORB.

— COM (Common Object Model) — технология, развиваемая корпорацией Microsoft на базе механизма OLE. Сетевой вариант этой технологии (для систем распределенных вычислений) известен под названием DCOM (Distributed COM). Объекты DCOM (в частности, объекты, которые можно вставлять в HTML-документы или к которым можно обращаться из Web-браузеров) известны под названием компонентов ActiveX. В COM/DCOM, как и в OpenDoc, можно использовать компоненты, написанные на разных объектно-ориентированных языках программирования. Но в отличие от OpenDoc в COM/DCOM остается естественная для Microsoft ориентация только на операционные системы Windows (реализация DCOM предусмотрена в ОС Windows NT 4.0). Технология ActiveX (прежнее название OLE Automation) обеспечивает интерфейс для управления объектами одного приложения из другого. В общем плане ActiveX — технология интеграции программного обеспечения фирмы Microsoft. Например, используя эту технологию, можно в среде VBA организовать доступ к объектам AutoCAD.

— JavaBeans — сравнительно новая технология, в которой используются компоненты, написанные на языке Java.

Рассмотрим подробнее основные CBD-технологии.

Организация связи клиента с серверными компонентами и в CORBA, и в DCOM происходит с помощью разновидностей языка IDL (Interface Definution Language). Язык IDL в CORBA позволяет описывать интерфейсы создаваемых компонентов. Описание, называемое метаданными, представляется в виде модуля, состоящего из заголовка, описаний типов данных, интерфейсов и операций. В заголовке указывается идентификатор модуля. В части типов данных перечисляются атрибуты, возвращаемые значения, исключительные ситуации. Примерами типов данных могут служить типы базовые (например, float, double, char, boolean, struct), конструируемые пользователем (например, записи и массивы) и объектные

ссылки, указывающие на интерфейсы компонентов. Описание интерфейсов начинается с ключевого слова interface, за которым следуют идентификаторы данного интерфейса и возможно наследуемых интерфейсов. Далее описываются операции (методы) в виде идентификаторов операций с возможными перечислениями параметров операций и указанием их принадлежности к входным или выходным данным.

Далее классы объектов (программные модули) должны быть реализованы в CORBA-среде. Для этого компилятор IDL выполняет следующие действия. Во-первых, метаданные для каждого класса объектов помещаются в специальную базу данных, имеющуюся в ORB, — репозитарий интерфейсов. Во-вторых, компилятор создает для каждого определенного на IDL метода клиентский и серверный стабы – специальные программные модули, обеспечивающие доступ к компонентам.

Основное назначение стабов – выполнение маршалинга и организация передачи данных через сеть. Маршалингом называют упаковку параметров в стандартный формат для пересылки. Маршалинг необходим по той причине, что представление данных в разных компьютерных средах может быть различным (например, различия в кодировке символов, в изображении чисел с плавающей запятой). Клиентский стаб будет использоваться для передачи вызовов и данных от клиента в сеть, а серверный стаб, называемый также скелетоном, будет вызывать метод уже в среде сервера и возвращать результаты.

На серверной стороне данные о каждом новом классе объектов, поддерживаемом конкретным сервером, заносятся в репозитарий реализаций. Эту операцию выполняет объектный адаптер. Обычно в ORB имеется несколько объектных адаптеров, обслуживающих разные группы компонентов (так, возможны объектные адаптеры, ориентированные на библиотеки, на базы данных, на группу отдельных программ и т.п.).

Объектные адаптеры выполняют также ряд других функций, например, таких как интерпретация объектных ссылок, активация и дезактивация компонентов, вызов их методов через скелетоны. Возможны разные способы активации и дезактивации компонентов. В первом из них для обслуживания каждого клиентского запроса создается своя копия компонента. В других способах копии не создаются, компонент обслуживает все запросы с разделением или без разделения во времени.

При реализации запроса брокер через объектный адаптер активирует соответствующий компонент. Далее клиент-серверное взаимодействие происходит через стабы.

Изложенную схему клиент-серверного взаимодействия называют статической. В CORBA предусмотрены также динамические вызовы. Для их осуществления не требуется предварительного формирования стабов с помощью компилятора языка IDL. Вместо стабов, специфических для каждого вызываемого метода, в CORBA используются специальные программы динамического взаимодействия, инвариантные к вызываемым методам. При этом необходимые данные для обращения к компоненту должны быть представлены в клиентской программе, в частности, они могут быть предварительно получены из репозитария интерфейсов. Динамические вызовы обеспечивают большую гибкость при программировании, но выполняются они значительно медленнее

В CORBA предусмотрен ряд унифицированных сервисов, работающих под управлением ORB. В частности, это сервисы:

— именования — присваивает объектам уникальные имена, в результате пользователь может искать объекты в сети по имени;

— жизненного цикла – обеспечивает создание, перемещение, копирование и удаление объектов (документов) в системе, в том числе составных объектов вместе со всеми ссылками и ассоциированными объектами;

— обработки транзакций — осуществляет управление транзакциями (блокировка, фиксация и откат транзакций) из приложений или из ОС, что позволяет многим объектам в сети использовать одни и те же серверы;

— событий — обеспечивает асинхронное распространение и обработку сообщений о событиях, происшедших при реализации процессов, что позволяет заинтересованным объектам координировать свои действия;

— обеспечения безопасности — поддерживает целостность данных.

В COM/DCOM все объекты сгруппированы в классы и каждый класс имеет свой идентификатор CLSID, а каждый интерфейс (метод) класса – свой идентификатор. Для создания объекта (экземпляра класса) клиент обращается к серверу библиотеки COM с указанием CLSID и идентификаторов всех требуемых интерфейсов. Сервер библиотеки COM находит в таблице-реестре по CLSID адрес удаленной машины, на которой размещен запрошенный компонент, и передает ей запрос клиента. На серверной стороне создается объект (копия компонента), он активируется и возвращает клиенту указатели-ссылки на требуемые интерфейсы. Теперь клиент может многократно обращаться к методам объекта, указывая в своих запросах имена интерфейса, методов и их параметров. Обычно объект исполняется на той машине, на которой размещен компонент, но можно выбрать и другую машину, указав в запросе, например, ее IP- адрес.

Появление технологии JavaBeans обусловлено успехом языка программирования Java. Технологию JavaBeans отличают от COM/DCOM две особенности. Во-первых, Java — единственный в JavaBeans язык программирования. Единственность языка и притом объектно-ориентированного обусловливает сравнительную легкость освоения и применения технологии JavaBeans. Во-вторых, технология JavaBeans является платформенно-независимой.

Компоненты в JavaBeans являются классами Java. Для их создания, модификации и объединения в прикладные программы имеются специальные средства в составе инструментальной среды JDK (Java Development Kit). С их помощью компоненты JavaBeans могут быть встроены в Java-аплеты, приложения или другие более крупные компоненты. В качестве Java-аплетов компоненты JavaBeans поддерживаются большинством имеющихся WWW-браузеров.

Развитие CBD-систем возможно в направлении дальнейшего упрощения программирования и, следовательно, сокращения сроков разработки ПО, однако это происходит за счет снижения степени универсальности соответствующих инструментальных средств. Такие более специализированные средства представляют собой группу компонентов, взаимосвязанных некоторым зависящим от приложения образом, и входят в системные среды САПР.

В общем случае компоненты системной среды объединены в несколько сценариев (потоков процедур или маршрутов), в которых выделяются точки входа для вставки специфичных пользовательских фрагментов и расширений. Имеются возможности не только вставки новых фрагментов, но и замены исходных компонентов в потоках процедур на оригинальные с сохранением интерфейса. Собственно многие системы, основанные на применении языков четвертого поколения (4GL), относятся именно к таким системным средам, в которых последовательности инкапсулированных модулей образуются с помощью операторов 4GL.

Пример реализации компонентно-ориентированной технологии в САПР. Основные идеи компонентно-ориентированной (объектной) технологии с созданием расширенных специализированных библиотек компонентов реализованы в системе CAS.CADE (Computer Aided Software/ Computer Aided Design Engineering) фирмы Matra Datavision.

Система CAS.CADE состоит из нескольких частей. Основными частями являются библиотеки классов и инструментальная среда для создания программного обеспечения (ПО) технических и научных приложений.

Библиотеки (Object Libraries) в CAS.CADE представляют собой специализированные наборы заранее разработанных компонентов на языке C++. Совокупность библиотек имеет иерархическую структуру. Базовые компоненты соответствуют классам объектной методологии. Примерами компонентов являются строки, списки, точки, матрицы, линии, поверхности, деревья, решатели уравнений, операторы сортировки, поиска на графах и т.п. Классы группируются в пакеты (Packages), пакеты – в наборы (Toolkits), наборы – в домены (Resourse Domains).

В CAS.CADE выделено несколько библиотек. Во-первых, это библиотеки 2D и 3D моделирования, включающие компоненты для определения, создания и манипулирования геометрическими моделями. Во-вторых, ряд библиотек предназначен для связи с ОС и управления данными, для обмена данными с внешними CAD системами, для создания сеточных моделей и др. Так, в состав библиотеки обмена данными входят конверторы данных из формата CAS.CADE в Express-файл прикладного протокола AP214 стандарта STEP и обратно. Аналогичные конверторы имеются для взаимного преобразования данных из формата CAS.CADE в другие популярные в САПР форматы IGES и DXF/SAT.

Необходимо отметить, что основные приложения, на которые ориентирована CAS.CADE, — это приложения машинной графики и геометрического моделирования, поэтому в системе наиболее развиты библиотеки графических и геометрических компонентов.

Геометрическое моделирование и визуализация в CAS.CADE поддерживаются соответствующим ПО. В это ПО входят библиотечные наборы “Геометрия”, “Топология”, “Визуализация” и др. Для тестирования и демонстрации компонентов перед их встраиванием в проектируемую прикладную САПР используются специальные язык, интерпретатор и просмотрщик, составляющие подсистему “Тестирование”.

Набор “Геометрия” включает пакеты канонических геометрических элементов и массивов (множеств) этих элементов.

Пакеты gp, geom2d и geom включают 2D и 3D геометрические элементы (классы), используемые в качестве сущностей в вычислительных процедурах, в том числе в таких операциях, как поворот, отражение, масштабирование и т.п. Примерами элементов могут служить декартовы координаты, точки, векторы, линии, окружности, квадратичные кривые, сферические, тороидальные и конические поверхности, кривые и поверхности Безье, B-сплайнов и др.

Большое число пакетов разработано для выполнения геометрических построений и метрических расчетов. Пакеты gce, GC, GCE2d включают алгоритмы построения сущностей из элементов пакетов gp, Geom, Geom2d, например, построение прямых, дуг окружностей, кривых по заданным параметрам таким, как инцидентные точки, центральные точки и радиусы, параллельные или нормальные прямые и т.п.

Набор “Топология” определяет структуры данных, описывающих связи (отношения) между геометрическими сущностями – классами предыдущего набора “Геометрия”. К структурам топологических данных относятся вершины, ребра, линии каркасных моделей, участки поверхности, оболочки – совокупности связанных через ребра участков поверхности, тела – части пространства, ограниченные оболочкой, совокупности тел, в том числе простые конструкции вида частей цилиндра, конуса, сферы, тора. В наборе имеются также средства: 1) для скругления острых углов и кромок, т.е. формирования галтелей постоянного или переменного радиуса; 2) для поддержания непрерывности при сопряжении разных поверхностей; 3) для метрических расчетов – определения длин ребер, площадей участков поверхности, объемов тел, центров масс и моментов инерции.

В подсистему “Тестирование” входят командный язык TCL (Test Command Language), на котором задается программа тестирования и просмотра библиотечных компонентов, интерпретатор TCL и 2D/3D визуализатор. В TCL имеются обычные для языков программирования команды, такие как присвоение значения переменной, организация цикла, условный переход, так и специальные команды. Среди последних выделяют базовые, геометрические и топологические команды. Примеры базовых команд: задержка при исполнении программы (например, при презентациях), обращение к файлу,

вывод на экран координат и других параметров геометрических объектов, создание окон для различных видов, масштабирование изображения, его поворот, установка цвета, выделение на экране одного заданного объекта и т.п. С помощью геометрических команд выполняют создание и модификацию кривых, поверхностей, геометрические преобразования типа поворота или зеркального отражения, вычисления координат, кривизн, производных, нахождение точек пересечения линий и поверхностей. Аналогичные действия производят по отношению к топологическим объектам с помощью топологических команд.

Инструментальная среда CAS.CADE включает интегрированную оболочку, подсистему проектирования пользовательского интерфейса, а также ряд многократно используемых специализированных программ, таких как 2D и 3D моделеры, подсистема управления данными, прикладные программы анализа и т.п.

Интегрированная оболочка служит для управления версиями и параллельной работой многих пользователей.

Для проектирования пользовательского интерфейса в CAS.CADE имеются специальные языковые и программные средства. Язык проектирования диалога состоит из команд создания интерфейса и доступа к компонентам.

Создание интерфейса включает создание контейнеров и диалоговых элементов. Контейнер представляет собой экранное окно, в котором будут размещаться элементы. Элементы обеспечивают информирование пользователя создаваемого приложения о возникающих событиях, дают возможность пользователю задавать значения параметров, выбирать режим работы и т.п.

Различают ряд видов контейнеров. Среди них контейнеры для сообщений, предупреждающих об ошибке, запрашивающих от пользователя ответы типа “да/нет”, задания размеров или цвета, выбора файла и т.п.

Примерами команд проектирования диалоговых элементов могут служить команды определения позиции элемента в окне, выбора одного элемента из заданного множества, конструирования текстовой строки или меню, фиксации событий, вызванных выбором мышью позиции или пункта меню, и др.

В структуре прикладной программы, создаваемой в среде CAS.CADE, можно выделить диалоговый модуль (модуль пользовательского интерфейса GUI – Graphic User Interface), модуль связи с прикладной частью и собственно прикладную часть, включающую отобранные компоненты и БД, зависящую от приложения

Объединение используемых в приложении компонентов в прикладную программу осуществляется на языке С++ или специальном языке описания интерфейсов, напоминающем язык IDL.. Следовательно, реализуются присущие С++ поддержка наследования и ограничение доступа (компоненты могут иметь статус защиты от несанкционированного доступа).

С помощью CAS.CADE создают специализированные приложения (прежде всего специализированные САПР) с сравнительно малыми затратами времени и средств.

Упражнения и вопросы для самоконтроля

1. Какие функции выполняет сетевое ПО?
2. Что понимают под менеджером и агентом в ПО управления сетью?
3. Что такое “эмуляция терминала”?
4. Охарактеризуйте различия между телеконференцией и видеоконференцией.
5. Назовите основные функции браузера.
6. Какие средства имеются в языке HTML для реализации гипертекста?
7. Что такое “электронная подпись”?
8. Перечислите основные особенности БнД в САПР.
9. Что такое “транзакция” в системах обработки данных?
10. Что понимают под системой PDM? Чем отличается система PDM от обычного БнД?
11. Назовите основные особенности хранилищ данных. Почему они используются в PDM?
12. Поясните механизм двухфазной фиксации транзакций в БнД.
13. В чем заключаются специфические особенности компонентно-ориентированных технологий разработки ПО?
14. Поясните назначение брокера ORB в технологии CORBA.
15. Что такое язык описания интерфейсов IDL?
16. Каковы назначение и структура системы CAS.CADE? Приведите примеры компонентов CAS.CADE.



Методики проектирования автоматизированных систем

6.1. Особенности проектирования автоматизированных систем

Этапы проектирования АС. К проектированию АС непосредственное отношение имеют два направления деятельности: 1) собственно *проектирование АС* конкретных предприятий (отраслей) на базе готовых программных и аппаратных компонентов с помощью специальных инструментальных средств разработки; 2) проектирование упомянутых *компонентов АС* и инструментальных средств, ориентированных на многократное применение при разработке многих конкретных автоматизированных систем.

Сущность первого направления можно охарактеризовать словами "*системная интеграция*" (другое близкое понятие имеет название — *консалтинг*). Разработчик АС должен быть специалистом в области системотехники, хорошо знать соответствующие международные стандарты, состояние и тенденции развития информационных технологий и программных продуктов, владеть инструментальными средствами разработки приложений (CASE-средствами) и быть готовым к восприятию и анализу автоматизируемых процессов в сотрудничестве с специалистами-прикладниками.

Существует ряд фирм, специализирующихся на разработке проектов АС (например, Price Waterhouse, Jet Info, Consistent Software, Interface и др.)

Второе направление в большей мере относится к области разработки математического и программного обеспечений для реализации функций АС — моделей, методов, алгоритмов, программ на базе знания системотехники, методов анализа и синтеза проектных решений, технологий программирования, операционных систем и т.п. Существует ряд общеизвестных технологий (методик) проектирования ПО АС, среди которых прежде всего следует назвать *компонентно-ориентированную разработку* — технологию индустриальной разработки программных систем CBD, поясненную в гл. 5.

Для каждого класса АС (САПР, АСУ, геоинформационные системы и т.д.) можно указать фирмы, специализирующиеся на разработке программных (а иногда и программно-аппаратных) систем. Многие из них на основе одной из базовых технологий реализуют свой подход к созданию АС и придерживаются стратегии либо тотального поставщика, либо открытости и расширения системы приложениями и дополнениями третьих фирм.

В России действует государственный стандарт на стадии создания автоматизированных систем ГОСТ 34.601-90. Существует и международный стандарт на стадии жизненного цикла программной продукции (ISO 12207:1995). Как собственно АС, так и компоненты АС являются сложными системами и при их проектировании можно использовать один из стилей проектирования:

— *нисходящее* (Top-of-Design); четкая реализация нисходящего проектирования приводит к *спиральной модели* разработки ПО, на каждом витке спирали блоки предыдущего уровня детализируются, используются обратные связи (альтернативой является так называемая *каскадная модель*, относящаяся к поочередной реализации частей системы);

— *восходящее* (Bottom-of-Design);

— *эволюционное* (Middle-of-Design).

Чаще всего используется нисходящий стиль блочно-иерархического проектирования.

Рассмотрим этапы нисходящего проектирования АС.

Верхний уровень проектирования АС часто называют *концептуальным* проектированием. Концептуальное проектирование выполняют в процессе предпроектных исследований, формулировки ТЗ, разработки эскизного проекта и прототипирования (согласно ГОСТ 34.601-90, эти стадии называют формированием требований к АС, разработкой концепции АС и эскизным проектом).

Предпроектные исследования проводят путем анализа (обследования) деятельности предприятия (компании, учреждения, офиса), на котором создается или модернизируется АС. При этом нужно получить ответы на вопросы: что не устраивает в существующей технологии? Что можно улучшить? Кому это нужно и, следовательно, каков будет эффект? Перед обследованием формируются и в процессе его проведения уточняются цели обследования — определение возможностей и ресурсов для повышения эффективности функционирования предприятия на основе автоматизации процессов уп-

равления, проектирования, документооборота и т.п. Содержание обследования — выявление структуры предприятия, выполняемых функций, информационных потоков, имеющихся опыта и средств автоматизации. Обследование проводят системные аналитики (системные интеграторы) совместно с представителями организации-заказчика.

На основе анализа результатов обследования строят модель, отражающую деятельность предприятия на данный момент (до реорганизации). Такую модель называют “As Is”. Далее разрабатывают исходную концепцию АС. Эта концепция включает в себя предложения по изменению структуры предприятия, взаимодействию подразделений, информационным потокам, что выражается в модели “To Be” (как должно быть).

Результаты анализа конкретизируются в ТЗ на создание АС. В ТЗ указывают потоки входной информации, типы выходных документов и предоставляемых услуг, уровень защиты информации, требования к производительности (пропускной способности) и т.п. ТЗ направляют заказчику для обсуждения и окончательного согласования.

Эскизный проект (техническое предложение) представляют в виде проектной документации, описывающей архитектуру системы, структуру ее подсистем, состав модулей. Здесь же содержатся предложения по выбору базовых программно-аппаратных средств, которые должны учитывать прогноз развития предприятия.

В отношении аппаратных средств и особенно ПО такой выбор чаще всего есть выбор фирмы-поставщика необходимых средств (или, по крайней мере, базового ПО), так как правильная совместная работа программ разных фирм достигается с большим трудом. В проекте может быть предложено несколько вариантов выбора. При анализе выясняются возможности покрытия автоматизируемых функций имеющимися программными продуктами и, следовательно, объемы работ по разработке оригинального ПО. Подобный анализ необходим для предварительной оценки временных и материальных затрат на автоматизацию. Учет ресурсных ограничений позволяет уточнить достижимые масштабы автоматизации, подразделить проектирование АС на работы первой, второй и т.д. очереди.

После принятия эскизного проекта разрабатывают *прототип* АС, представляющий собой набор программ, эмулирующих работу готовой системы. Благодаря прототипированию можно не только разработчикам, но и будущим пользователям АС увидеть контуры и особенности системы и, следовательно, заблаговременно внести коррективы в проект.

Как на этапе обследования, так и на последующих этапах целесообразно придерживаться определенной дисциплины фиксации и представления получаемых результатов, основанной на той или иной методике формализации спецификаций. Формализация нужна для однозначного понимания исполнителями и заказчиком требований, ограничений и принимаемых решений.

При концептуальном проектировании применяют ряд спецификаций, среди которых центральное место занимают *модели* преобразования, хранения и передачи информации. Модели, полученные в процессе обследования предприятия, являются моделями его функционирования. В процессе разработки АС модели, как правило, претерпевают существенные изменения (переход от “As Is” к “To Be”) и в окончательном виде модель “To Be” рассматривают в качестве модели проектируемой АС.

Различают функциональные, информационные, поведенческие и структурные модели. *Функциональная* модель системы описывает совокупность выполняемых системой функций. *Информационные* модели отражают структуры данных — их состав и взаимосвязи. *Поведенческие* модели описывают информационные процессы (динамику функционирования), в них фигурируют такие категории, как состояние системы, событие, переход из одного состояния в другое, условия перехода, последовательность событий, осуществляется привязка ко времени. *Структурные* модели характеризуют морфологию системы (ее построение) — состав подсистем, их взаимосвязи.

Содержанием последующих этапов нисходящего проектирования (согласно ГОСТ 34.601-90, это стадии разработки технического проекта, рабочей документации, ввода в действие) является уточнение перечней приобретаемого оборудования и готовых программных продуктов, построение системной среды, детальное инфологическое проектирование БД и их первоначального наполнения, разработка собственного оригинального ПО, которая, в свою очередь, делится на ряд этапов нисходящего проектирования. Эти работы составляют содержание *рабочего проектирования*. После этого следуют

закупка и инсталляция программно-аппаратных средств, внедрение и опытная эксплуатация системы.

Особое место в ряду проектных задач занимает разработка проекта корпоративной вычислительной сети, поскольку техническое обеспечение АС имеет сетевую структуру.

Если территориально АС располагается в одном здании или в нескольких близко расположенных зданиях, то корпоративная сеть может быть выполнена в виде совокупности нескольких локальных подсетей типа Ethernet или Token Ring, связанных опорной локальной сетью типа FDDI или высокоскоростной Ethernet. Кроме выбора типов подсетей, связных протоколов и коммутационного оборудования приходится решать задачи распределения узлов по подсетям, выделения серверов, выбора сетевого ПО, определения способа управления данными в выбранной схеме распределенных вычислений и т.п.

Если АС располагается в удаленных друг от друга пунктах, в частности, расположенных в разных городах, то решается вопрос об аренде каналов связи для корпоративной сети, поскольку альтернативный вариант использования выделенного канала в большинстве случаев оказывается неприемлемым по причине высокой цены. Естественно, что при этом прежде всего рассматривается возможность использования услуг Internet. Возникающие при этом проблемы связаны с обеспечением информационной безопасности и надежности доставки сообщений.

Рекомендации по проектированию корпоративных сетей. Основные сетевые протоколы и технологии реализованы в программных и аппаратных средствах ряда фирм, и задача проектировщика сети (системного интегратора) — правильно выбрать эти средства для заданных условий конкретного предприятия, обеспечив требуемый уровень производительности и надежности при минимизации затрат.

Среди основных рекомендаций следует упомянуть следующие.

1. Информатизацию и автоматизацию деятельности предприятия необходимо начинать с анализа процессов функционирования его подразделений. Следует выявить информационные потребности подразделений, решаемые задачи, информационные потоки между подразделениями, установить, какие процессы требуют автоматизации и компьютеризации и в какую очередь. Целесообразно проводить эту работу совместно с работниками самих подразделений, с самого начала выделить сотрудников предприятия, которые будут поддерживать и развивать информационную структуру, вычислительные и сетевые средства.

2. Если сеть создается заново (особенно в новых зданиях), целесообразен комплексный подход к проектированию кабельной системы сети. При этом в проекте нужно учитывать прокладку не только коммуникаций для передачи данных, но и одновременно соединений телефонной связи, проводов пожарной и охранной сигнализации, кабельного телевидения и т.п., а возможно, и использование для этих целей некоторых общих кабельных соединений.

3. При выборе типа линий связи между отдельно стоящими зданиями необходимо провести сравнительный анализ проводных линий и радиоканалов.

4. В наиболее популярном варианте кабельной системы и размещения коммутационного оборудования внутри здания рекомендуется под коммутационное оборудование отводить помещение на этаже с максимальным числом рабочих мест, горизонтальную (этажную) проводку выполнять витой парой категории 5 (длина до 90 м) или коаксиальным кабелем, вертикальную проводку (межэтажную) — ВОЛС или коаксиальным кабелем.

5. Относительно выбора одного из двух наиболее популярных вариантов построения подсетей (ЛВС) — Ethernet или Token Ring однозначные выводы отсутствуют. Если нагрузка подсети может превышать 35% (т.е. без учета конфликтов передача данных в сети занимает 35% времени), то лучше использовать Token Ring. При меньшей загрузке предпочтительнее Ethernet, так как обеспечиваются меньшие задержки. Вариант Ethernet можно применять и при большем трафике, но тогда нужно предусмотреть разделение ЛВС на подсети с мостовым соединением между ними. Следует также рассмотреть целесообразность использования виртуальных ЛВС.

6. Как сказано выше, при выборе типов коммутационного оборудования полезно ориентироваться на средства, предоставляемые одной фирмой, иначе возможны нестыковки, несмотря на общность используемых стандартов, могут возникнуть затруднения при последующей эксплуатации и развитии сети.

7. Если сеть связывает удаленные друг от друга здания, в частности, расположенные в разных городах, то возможны варианты использования выделенных каналов связи или сетей общего пользования (прежде всего Internet). Второй вариант обходится значительно дешевле, но в этом случае нужно обратить особое внимание на обеспечение информационной безопасности (разграничение доступа, установка защитных экранов — брандмауэров и т.п.).

8. Для корректировки и верификации проекта сети нужно использовать имеющиеся средства имитационного моделирования.

Примерами программ анализа и моделирования вычислительных сетей могут служить COMNET III и OPNET. Ниже приведены краткие характеристики этих программ.

COMNET III; (фирма CACI Products Company; http://www.caciasl.com) выполняет интерактивное моделирование работы локальных и территориальных вычислительных сетей. Исходные данные задаются на проблемно-ориентированных языках моделирования MODSIM или SIMSCRIPT с графическими расширениями. На экране ЭВМ изображается топология сети с указанием узлов, линий связи, источников данных (трафика). В результате моделирования определяются “узкие” места, задержки в передаче данных, загрузка линий, буферов, процессоров, длины очередей, пиковые нагрузки. Имеется библиотека моделей протоколов и аппаратных средств: маршрутизаторов (3COM, Cisco, DEC, HP и др.), алгоритмов протоколов (TCP/IP, SNA, RIP, OSPF, IGRP и др.) и ряда методов доступа (CSMA/CD, FDDI, ALOHA).

OPNET (Planner and Modeler); (фирма OPNET; http://www.mil3.com) выполняет анализ работы различных локальных и территориальных гетерогенных вычислительных сетей, в том числе высокоскоростных сетей FDDI и ATM, радиоканалов с временным мультиплексированием и др. На входном графическом языке задается структура сетей с указанием процессоров, источников потоков данных, очередей, трансмиттеров и т.п. Система позволяет сравнивать различные архитектуры построения сетей, определять размещение серверов, рассчитывать трафик. В библиотеке системы имеются модели различных протоколов (Ethernet, FDDI, TCP/IP, ATM, PSTN, Frame Relay и др.).

Математическое обеспечение для моделирования сетей и сетевых протоколов — системы массового обслуживания и (или) сети Петри. Для структурного синтеза сетей используют дискретное математическое программирование и экспертные системы, перспективно применение генетических алгоритмов синтеза. Существуют пакеты интерактивного проектирования сетей. С их помощью можно изобразить поэтажную схему здания, разместить на ней обозначения компьютеров и сетевого оборудования, выбрать из базы данных типы оборудования и каналов связи, проверить допустимость их совместного использования и другие ограничения. Пример такого пакета — NetSuit Advanced Professional Design фирмы NetSuit Development.

9. Разрабатывается конфигурация сети. Все узлы сети распределяются по рабочим группам, а затем рабочие группы — по подсетям. Исходя из оценок прогнозируемого трафика и его характера, числа узлов и подсетей выбирается структура сети и типы сетевого оборудования. Если нет уверенности в том, что состав пользователей в рабочих группах будет стабильным, то целесообразно использовать виртуальные ЛВС. Необходимо учитывать возможности масштабирования сети, если ожидается ее расширение в процессе эксплуатации.

Обеспечение открытости автоматизированных систем. Одной из главных тенденций современной индустрии информатики является создание *открытых систем*. Свойство открытости означает, во-первых, переносимость (мобильность) ПО на различные аппаратные платформы, во-вторых, приспособленность системы к ее модификациям (модифицируемость или собственно открытость) и комплексированию с другими системами с целью расширения ее функциональных возможностей и (или) придания системе новых качеств (интегрируемость).

Переход к открытым информационным системам позволяет существенно ускорить научно-технический прогресс в результате замены длительной и дорогостоящей разработки новых систем по полному циклу их компоновкой из ранее спроектированных подсистем или быстрой модернизацией уже существующих систем (реинжиниринг) .

Открытость подразумевает выделение в системе интерфейсной части (входов и выходов), обеспечивающей сопряжение с другими системами или подсистемами, причем для комплексирования достаточно располагать сведениями только об интерфейсных частях сопрягаемых объектов. Если же интерфейсные части выполнены в соответствии с заранее оговоренными правилами и соглашениями, которых должны придерживаться все создатели открытых систем определенного приложения, то проблема создания новых сложных систем существенно упрощается. Из этого следует, что основой создания открытых систем является стандартизация и унификация в области информационных технологий.

Значительное развитие концепция открытости получила в области построения вычислительных сетей, что нашло выражение в эталонной модели взаимосвязи открытых систем, поддерживаемой рядом международных стандартов. Идеи открытости широко используются при построении программного, информационного и лингвистического обеспечений автоматизированных систем; в результате повышается степень универсальности программ и расширяются возможности их адаптации к конкретным условиям.

Аспекты открытости выражаются в стандартизации:

— API (Application Program Interface) — интерфейсов прикладных программ с операционным окружением, в том числе системных вызовов и утилит ОС, т.е. связей с ОС;

— межпрограммного интерфейса, включая языки программирования;

— сетевого взаимодействия;

— пользовательского интерфейса, в том числе средств графического взаимодействия пользователя с ЭВМ;

— средств защиты информации.

Стандарты, обеспечивающие открытость ПО, в настоящее время разрабатываются такими организациями, как ISO (International Standard Organization), IEEE (Institute of Electrical and Electronics Engineers), EIA (Electronics Industries Association) и рядом других.

Выше уже были отмечены телекоммуникационные и сетевые стандарты семиуровневой модели взаимосвязи открытых систем (ЭМВОС).

Стандарты POSIX (Portable Operating System Interface) предназначены для API и составляют группу стандартов IEEE 1003. В этих стандартах содержатся перечень и правила вызова интерфейсных функций, определяются способы взаимодействия прикладных программ с ядром ОС на языке С (что означает преимущественную ориентацию на ОС Unix), даны расширения для взаимодействия с программами на других языках, способы тестирования интерфейсов на соответствие стандартам POSIX, правила административного управления программами и данными и т.п.

Ряд стандартов ISO посвящен языкам программирования. Имеются стандарты на языки С (ISO 9899), Фортран (ISO 1539), Паскаль (ISO 7185) и др.

Среди других стандартов, способствующих открытости ПО АС, следует отметить стандарты графического пользовательского интерфейса, хранения и передачи графических данных, построения БД и файловых систем, сопровождения и управления конфигурацией программных систем и др.

Важное значение для создания открытых систем имеет унификация и стандартизация средств межпрограммного интерфейса или, другими словами, необходимо наличие профилей АС для информационного взаимодействия программ, входящих в АС. *Профилем* открытой системы называют совокупность стандартов и других нормативных документов, обеспечивающих выполнение системой заданных функций.

Так, в профилях АС могут фигурировать язык EXPRESS стандарта STEP, спецификация графического пользовательского интерфейса Motif, унифицированный язык SQL обмена данными между различными СУБД, стандарты сетевого взаимодействия, в профили САПР машиностроения может входить формат IGES и в случае САПР радиоэлектроники — формат EDIF и т.п.

Всего в информационных технологиях уже к 1997 г. было более 1000 стандартов. Профили создаются для их упорядочения, получения взаимоувязанных целостных совокупностей для построения конкретных систем.

Например, предлагаются профили AMH11 передачи сообщений между прикладными и транспортным уровнями; TA51 — устанавливает требования к работе оконечной системы в IEEE 802.3, RA51.1111 — ретрансляция услуг сетевого уровня между МДКН/ОК и PSDN (Packed Switched Data Network) и др. Теперь можно выбрать один базовый стандарт и соответствующее средство выдаст профиль — все остальные необходимые стандарты.

6.2. Инструментальные средства концептуального проектирования

CASE-системы. В современных информационных технологиях важное место отводится инструментальным средствам и средам разработки АС, в частности, системам разработки и сопровождения их ПО. Эти технологии и среды образуют системы, называемые *CASE-системами*.

Используется двоякое толкование аббревиатуры CASE, соответствующее двум направлениям использования CASE-систем. Первое из них — Computer Aided System Engineering — подчеркивает направленность на поддержку концептуального проектирования сложных систем, преимущественно слабоструктурированных. Далее CASE-системы этого направления будем называть *системами CASE для концептуального проектирования*. Второе направление было рассмотрено выше, его название

Computer Aided Software Engineering переводится, как автоматизированное проектирование программного обеспечения, соответствующие CASE-системы называют *инструментальными CASE* или *инструментальными средами* разработки ПО (одно из близких к этому названий — RAD — Rapid Application Development).

Среди систем CASE для концептуального проектирования различают системы функционального, информационного или поведенческого проектирования. Наиболее известной методикой *функционального проектирования* сложных систем является методика SADT (Structured Analysis and Design Technique), предложенная в 1973 г. Р.Россом и впоследствии ставшая основой международного стандарта IDEF0 (Integrated DEFinition 0).

Системы *информационного проектирования* реализуют методики инфологического проектирования БД. Широко используются язык и методика создания информационных моделей приложений, закрепленные в международном стандарте IDEF1X. Кроме того, развитые коммерческие СУБД, как правило, имеют в своем составе совокупность CASE-средств проектирования приложений.

Основные положения стандартов IDEF0 и IDEF1X использованы также при создании комплекса стандартов ISO 10303, лежащих в основе технологии STEP для представления в компьютерных средах информации, относящейся к проектированию и производству в промышленности.

Поведенческое моделирование сложных систем используют для определения динамики функционирования сложных систем. В его основе лежат модели и методы имитационного моделирования систем массового обслуживания, сети Петри, возможно применение конечно-автоматных моделей, описывающих поведение системы, как последовательности смены состояний.

Применение инструментальных CASE-систем ведет к сокращению затрат на разработку ПО за счет уменьшения числа итераций и числа ошибок, к улучшению качества продукта за счет лучшего взаимопонимания разработчика и заказчика, к облегчению сопровождения готового ПО.

Среди инструментальных CASE-систем различают интегрированные комплексы инструментальных средств для автоматизации всех этапов жизненного цикла ПО (такие системы называют Workbench) и специализированные инструментальные средства для выполнения отдельных функций (Tools). Средства CASE по своему функциональному назначению принадлежат к одной из следующих групп: 1) средства программирования; 2) средства управления программным проектом; 3) средства верификации (анализа) программ; 4) средства документирования.

К первой группе относятся компиляторы с алгоритмических языков; построители диаграмм потоков данных; планировщики для построения высокоуровневых спецификаций и планов ПО (возможно на основе баз знаний, реализованных в экспертных системах); интерпретаторы *языков спецификаций* и *языков четвертого поколения*; прототайпер для разработки внешних интерфейсов — экранов, форм выходных документов, сценариев диалога; генераторы программ определенных классов (например, конверторы заданных языков, драйверы устройств программного управления, постпроцессоры); кросс-средства; отладчики программ. При этом под *языками спецификаций* понимают средства укрупненного описания разрабатываемых алгоритмов и программ, к языкам 4GL относят языки для компиляции программ из набора готовых модулей, реализующих типовые функции достаточно общих приложений (чаще всего это функции технико-экономических систем).

Управление программным проектом называют также *управлением конфигурациями* ПО (SCM — software configuration management). Этому понятию соответствуют корректное внесение изменений а программную систему при ее проектировании и сопровождении, контроль целостности проектных данных, управление версиями проекта, организация параллельной работы членов коллектива разработчиков. Использование средств управления конфигурациями позволяет создавать программные системы из сотен и тысяч модулей, значительно сокращать сроки разработки, успешно модернизировать уже поставленные заказчикам системы.

Основой средств управления программным проектом является репозиторий — БД проекта. Именно в репозитории отражена история развития программного проекта, содержатся все созданные версии (исходный программный код, исполняемые программы, библиотеки, сопроводительная документация и т.п.) с помощью репозитория осуществляется контроль и отслеживание вносимых изменений.

Средства верификации служат для оценки эффективности исполнения разрабатываемых программ и определения наличия в них ошибок и противоречий. Различают статические и динамические анализаторы. В статических анализаторах ПО исследуется на наличие неопределенных данных, бесконечных циклов, недопустимых передач управления и т.п. Динамический анализатор функционирует в процессе исполнения проверяемой программы; при этом исследуются трассы, измеряются частоты обращений к модулям и т.п. Используемый математический аппарат — сети Петри, теория массового обслуживания.

В последнюю из перечисленных групп входят документаторы для оформления программной документации, например, отчетов по данным репозитория; различные редакторы для объединения, разделения, замены, поиска фрагментов программ и других операций редактирования.

Проектирование ПО с помощью CASE-систем включает в себя несколько этапов. Начальный этап — предварительное изучение проблемы. Результат представляют в виде исходной диаграммы потоков данных и согласуют с заказчиком. На следующем этапе выполняют детализацию ограничений и функций программной системы, и полученную логическую модель вновь согласуют с заказчиком. Далее разрабатывают физическую модель, т.е. определяют модульную структуру программы, выполняют инфологическое проектирование БД, детализируют граф-схемы программной системы и ее модулей.

Спецификации проектов программных систем. Важное значение в процессе разработки ПО имеют средства спецификации проектов ПО. Средства спецификации в значительной мере определяют суть методов CASE.

Способы и средства спецификации классифицируют по базовой методологии, используемой для декомпозиции ПО, как сложной системы, и по аспектам моделирования ПО.

Различают два подхода к декомпозиции ПО Первый способ называют *функциональным* или *структурным*. Он основан на выделении функций и потоков данных. Второй способ – *объектный*, выражает идеи объектно-ориентированного проектирования и программирования. Проектирование ПО из готовых компонентов, рассмотренное в предыдущей главе, есть выражение объектного подхода.

Аспектами моделирования приложений являются функциональное, поведенческое и информационное описания.

Практически все способы *функциональных* спецификаций имеют следующие общие черты:

— модель имеет иерархическую структуру, представляемую в виде диаграмм нескольких уровней;

— элементарной частью диаграммы каждого уровня является конструкция вход-функция-выход;

— необходимая дополнительная информация содержится в файлах поясняющего текста.

В большинстве случаев функциональные диаграммы являются диаграммами потоков данных (DFD — Data Flow Diagram). В DFD блоки (прямоугольники) соответствуют функциям, дуги — входным и выходным потокам данных. Поясняющий текст представлен в виде “словарей данных”, в которых указаны компонентный состав потоков данных, число повторений циклов и т.п. Для описания структуры информационных потоков можно использовать нотацию Бэкуса-Наура.

Одна из нотаций для DFD предложена Е.Йорданом. В ней описывают процессы (функции), потоки данных, хранилища и внешние сущности, их условные обозначения показаны на рис. 6.1.

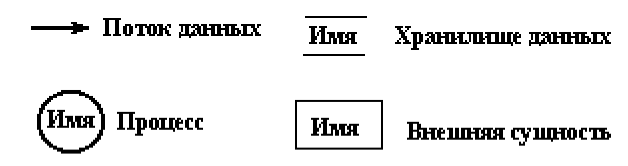


Рис. 6.1. Изображения элементов в нотации Йордана

Разработка DFD начинается с построения диаграммы верхнего уровня, отражающей связи программной системы, представленной в виде единого процесса, с внешней средой. Декомпозиция процесса проводится до уровня, на котором фигурируют элементарные процессы, которые могут быть представлены одностраничными описаниями алгоритмов (миниспецификациями) на терминальном языке программирования.

Для описания *информационных* моделей наибольшее распространение получили диаграммы сущность-связь (ERD — Entity-Relation Diagrams), в которых предусмотрены средства для описания сущностей, атрибутов и отношений. Спецификации хранилищ данных в CASE, как правило, даются с помощью диаграмм сущность-связь Стандартной методикой построения таких диаграмм является IDEF1X.

Поведенческие модели описывают процессы обработки информации. В инструментальных CASE-системах их представляют в виде граф-схем, диаграмм перехода состояний, таблиц решений, псевдокодов (языков спецификаций), процедурных языков программирования, в том числе языков четвертого поколения.

В граф-схемах блоки, как и в DFD, используют для задания процессов обработки, но дуги имеют иной смысл — они описывают последовательность передач управления (вместе с специальными блоками управления).

В диаграммах перехода состояний узлы соответствуют состояниям моделируемой системы, дуги — переходам из состояние в состояние, атрибуты дуг — условиям перехода и инициируемым при их выполнении действиям. Очевидно, что как и в других конечно-автоматных моделях, кроме графической формы представления диаграмм перехода состояний, можно использовать также табличные формы. Так, при изоморфном представлении с помощью таблиц перехода состояний каждому переходу соответствует строка таблицы, в которой указываются исходное состояние, условие перехода, инициируемое при этом действие и новое состояние после перехода.

Близкий по своему характеру способ описания процессов основан на таблицах (или деревьях) решений. Каждый столбец таблицы решений соответствует определенному сочетанию условий, при выполнении которых осуществляются действия, указанные в нижерасположенных клетках столбца.

Таблицы решений удобны при описании процессов с многократными ветвлениями. В этих случаях помогают также визуальные языки программирования, в которых для описания процессов используют графические элементы, подобные приведенным на рис. 6.2.

В псевдокодах алгоритмы записываются с помощью как средств некоторого языка программирования (преимущественно для управляющих операторов), так и естественного языка (для выражения содержания вычислительных блоков). Используются конструкции (операторы) следования, условные, цикла. Служебные слова из базового языка программирования или из DFD записываются заглавными буквами, фразы естественного языка — строчными.

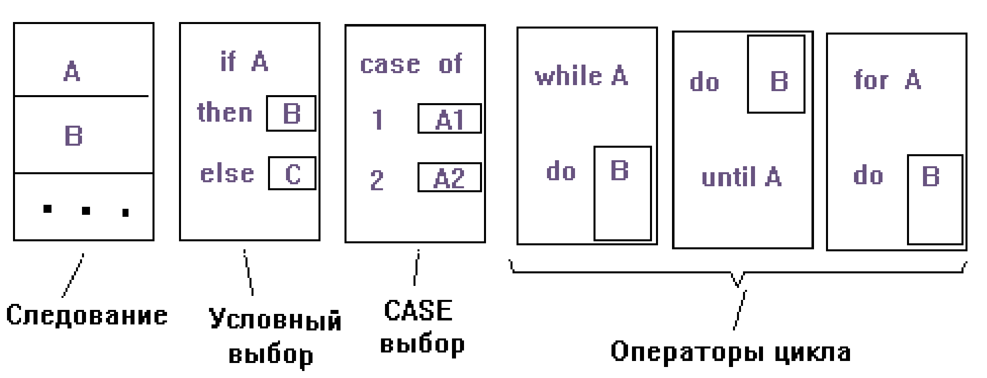


Рис. 6.2. Примеры описания операторов в визуальных языках программирования

Языки четвертого поколения предназначены для описания программ как совокупностей заранее разработанных программных модулей. Поэтому одна команда языка четвертого поколения может соответствовать значительному фрагменту программы на языке 3GL. Примерами языков 4GL могут служить Informix-4GL, JAM, NewEra, XAL.

Миниспецификации процессов могут быть выражены с помощью псевдокодов (языков спецификаций), визуальных языков проектирования или языков программирования,

Объектный подход представлен компонентно-ориентированнными технологиями разработки ПО. При объектном подходе ПО формируется из компонентов, объединяющих в себе алгоритмы и данные и взаимодействующих путем обмена сообщениями. Для поддержки объектного подхода разработан стандартный язык моделирования приложений UML.

Технологии реинжиниринга и параллельного проектирования. Взаимосвязанная совокупность методик IDEF для концептуального проектирования разработана по программе Integrated Computer Aided Manufacturing в США. В этой совокупности имеются методики функционального, информационного и поведенческого моделирования и проектирования, в ее состав в настоящее время входят IDEF-методики, представленные в приложении (табл. П.1), часть из которых имеет статус международного стандарта.

Методики IDEF задают единообразный подход к моделированию приложений, но не затрагива-

ют проблем единообразного представления данных в процессах информационного обмена между разными компьютерными системами и приложениями. Необходимость решения этих проблем в интегрированных АС привела к появлению ряда унифицированных форматов представления данных в межкомпьютерных обменах, среди которых наиболее известными являются форматы IGES, DXF (в машиностроительных приложениях), EDIF (в электронике) и некоторые другие. Однако ограниченные возможности этих форматов обусловили продолжение работ в направлении создания более совершенных методик и представляющих их стандартов. На эту роль в настоящее время претендует совокупность стандартов STEP.

Методика IDEF0. Как отмечено выше, наиболее известной методикой *функционального моделирования* сложных систем является методика SADT (Structured Analysis and Design Technique), положенная в основу стандарта IDEF0.

IDEF0 — это более четко очерченное представление методики SADT. SADT — методика, рекомендуемая для начальных стадий проектирования сложных искусственных систем управления, производства, бизнеса, включающих людей, оборудование, ПО. Начиная с момента создания первой версии, методика успешно применялась для проектирования телефонных сетей, систем управления воздушными перевозками, производственных предприятий и др.

Разработку SADT-модели начинают с формулировки вопросов, на которые модель должна давать ответы, т.е. формулируют цель моделирования. Далее строят иерархическую совокупность диаграмм с лаконичным описанием функций.

Недостатки SADT-моделей — их слабая формализованность для автоматического выполнения проектных процедур на их основе. Однако наличие графического языка диаграмм, удобного для восприятия человеком, обусловливает полезность и применимость методики SADT.

Описание объектов и процессов в SADT (IDEF0) выполняется в виде совокупности взаимосвязанных блоков (рис. 6.3).

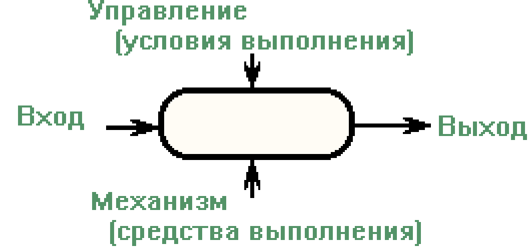


Рис. 6.3. Блок ICOM в IDEF0-диаграммах

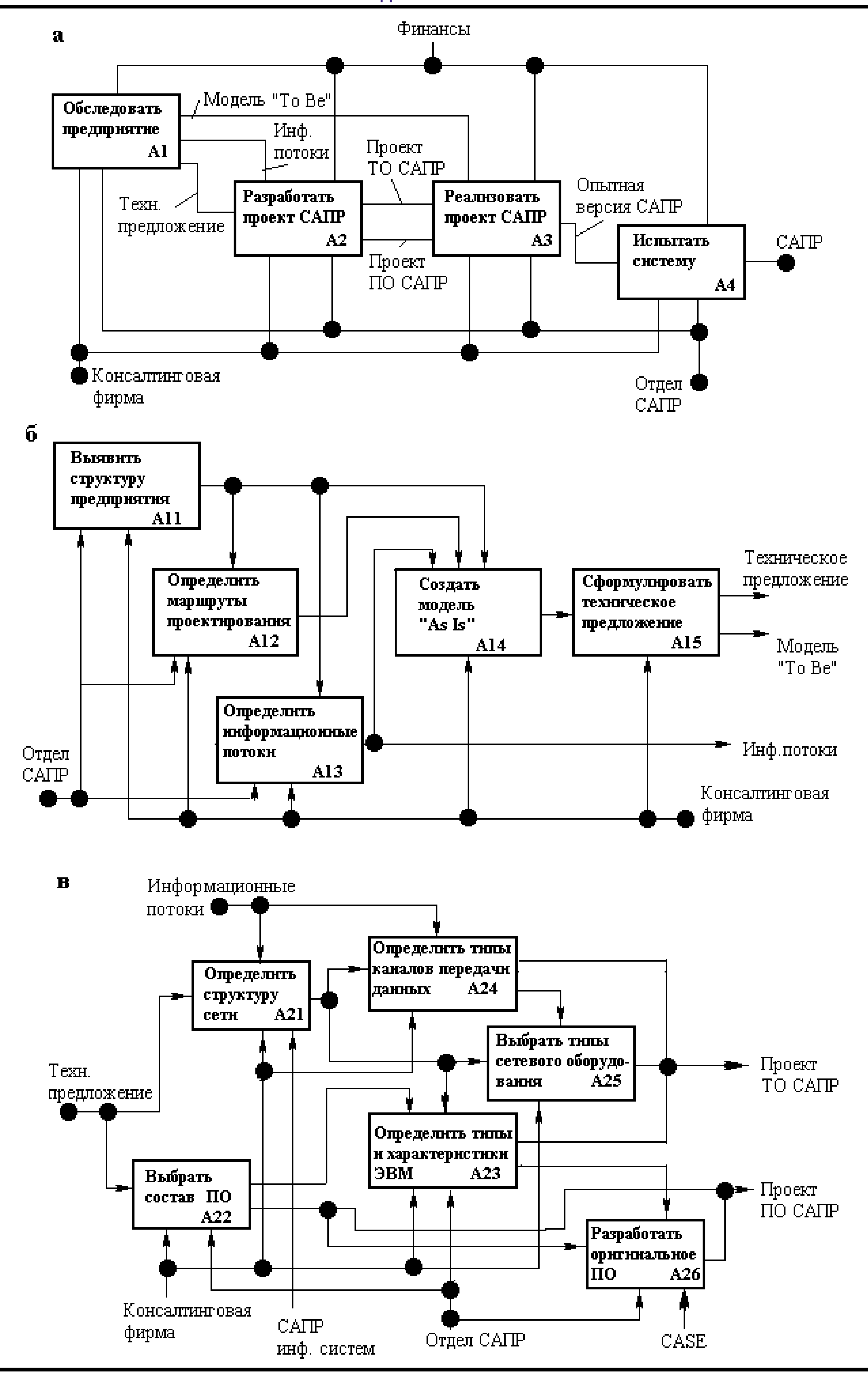
Блоки выражают функции (работы), поэтому их названиями обычно являются глаголы или отглагольные существительные. Типичные примеры функций: планировать, разработать, классифицировать, измерить, изготовить, отредактировать, рассчитать, продать (или планирование, разработка, классификация, измерение, изготовление, редактирование, расчет, продажа). Число блоков на одном уровне иерархии — не более 6, иначе восприятие диаграмм будет затруднено. Число уровней иерархии не ограничено, но обычно их не более 5. Блоки нумеруются (номер записывается в правом нижнем углу). Дуги (стрелки) отображают множества объектов (данных), их имена — существительные. Управление определяет условия выполнения, примеры управления: требования, чертеж, стандарт, указания, план. Механизм выражает используемые средства, например: компьютер, оснастка, заказчик, фирма. Входы и выходы могут быть любыми объектами.

Блоки рис. 6.3 в англоязычной литературе называют блоками ICOM (Input — Control — Output — Mechanism).

Рассмотрим пример функциональной модели для процесса создания САПР на предприятии, на котором ранее автоматизация проектирования была развита слабо.

Диаграмма верхнего (нулевого) уровня А0 включает единственный блок ICOM “Разработать САПР”. В качестве исполнителей фигурируют специализированная организация, занимающаяся проектированием автоматизированных систем и называемая консалтинговой фирмой, а также представители организации-заказчика, объединенные в создаваемый на предприятии отдел САПР.

Диаграмма первого уровня, показанная на рис. 6.4,а, включает блоки А1 — обследования предприятия, А2 — проектирования САПР, А3 — реализации САПР и А4 – испытаний системы. Диаграммы следующего второго уровня, раскрывающие первые блоки А1, А2 и А3, представлены на рис. 6.4,б, в и г соответственно (на этих рисунках не отмечены данные, соответствующие внутренним стрелкам диаграмм). При обследовании предприятия специалисты консалтинговой фирмы вместе с работниками отдела САПР, изучают структуру предприятия, типичные маршруты проектирования, информационные потоки и на этой базе разрабатывают модель “As Is”. Далее создается новая модель “To Be” с учетом не только требований автоматизации проектирования, но и будущих информационных потребностей процессов управления и делопроизводства. Модель “To Be” составляет основу технического предложения на создание САПР.


а
Финансы
Обследовать предприятие A1
Модель "То Be"
Инф. потоки
Проект ТО САПР
Техн. предложение
Разработать проект САПР A2
Реализовать проект САПР A3
Опытная версия САПР
Испытать систему A4
САПР
Проект ПО САПР
Консалтинговая фирма
Отдел САПР
б
Выявить структуру предприятия A11
Определить маршруты проектирования A12
Создать модель "As Is" A14
Сформулировать техническое предложение A15
Техническое предложение
Модель "То Be"
Определить информационные потоки A13
Инф.потоки
Отдел САПР
Консалтинговая фирма
в
Информационные потоки
Определить типы каналов передачи данных A24
Определить структуру сети A21
Выбрать типы сетевого оборудо-вания A25
Проект ТО САПР
Техн. предложение
Определить типы и характеристики ЭВМ A23
Выбрать состав ПО A22
Проект ПО САПР
Разработать оригинальное ПО A26
Консалтинговая фирма
САПР инф. систем
Отдел САПР
CASE

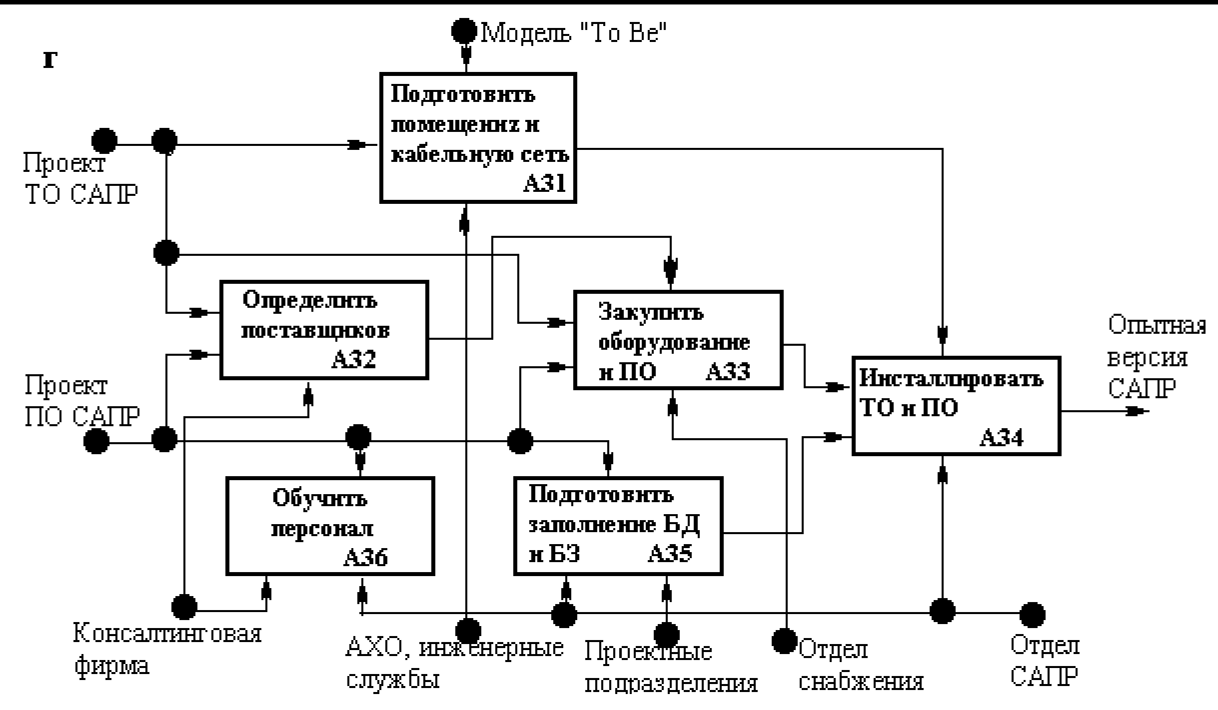


Рис. 6.4. Функциональная модель процесса создания САПР:
а) IDEF0-диаграмма первого уровня; **б)** IDEF0-диаграмма обследования предприятия;
в) IDEF0-диаграмма проектирования САПР; **г)** IDEF0-диаграмма реализации проекта САПР

При проектировании САПР выбирают аппаратно-программную платформу, базовое ПО проектирующих и обслуживающих подсистем, разрабатывают структуру корпоративной сети, определяют типы сетевого оборудования, серверов и рабочих станций, выявляют необходимость разработки оригинальных программных компонентов.

Реализация проекта САПР включает подготовку помещений, монтаж кабельной сети, обучение будущих пользователей САПР, закупку и инсталляцию ТО и ПО.

Разработка SADT-моделей состоит из ряда этапов.

1. Сбор информации. Источниками информации могут быть документы, наблюдение, анкетирование и т.п. Существуют специальные методики выбора экспертов и анкетирования.

2. Создание модели. Используется нисходящий стиль: сначала разрабатываются верхние уровни, затем нижние.

3. Рецензирование модели. Реализуется в итерационной процедуре рассылки модели на отзыв и ее доработки по замечаниям рецензентов, в завершение собирается согласительное совещание.

Связи функциональной модели, отражающей функции, со структурной моделью, отражающей средства выполнения функций, выражаются с помощью специальных словарей, дающих однозначное толкование вводимым именам ресурсов.

Дальнейшее использование IDEF0-модели — конкретизация задач выбора ресурсов, разработка планов реализации, переход к имитационным моделям и т.п.

Методика IDEF3. *Поведенческое моделирование* сложных систем используют для исследования динамики их функционирования. В основе поведенческого моделирования лежат модели и методы имитационного моделирования систем массового обслуживания, сети Петри, возможно применение конечно-автоматных моделей, описывающих поведение системы, как последовательности смены состояний.

Поведенческие аспекты приложений отражает методика IDEF3. Если методика IDEF0 связана с функциональными аспектами и позволяет отвечать на вопросы “Что делает система?”, то в IDEF3 детализируются и конкретизируются IDEF0-функции, IDEF3-модель отвечает на вопросы “Как система это делает?” Язык IDEF3 — язык диаграмм, помогающий разработчику моделей наглядно представить моделируемые процессы. В IDEF3 входят два типа описаний: 1) процесс-ориентированные в виде последовательности операций; 2) объект-ориентированные, выражаемые диаграммами перехода состояний, характерными для конечно-автоматных моделей.

На рис. 6.5 представлен пример процесс-ориентированной IDEF3-диаграммы. Здесь функции (операции) показаны прямоугольниками с горизонтальной чертой, отделяющей верхнюю секцию с названием функции от нижней секции, содержащей номер функции. Связи, отражающие последова-

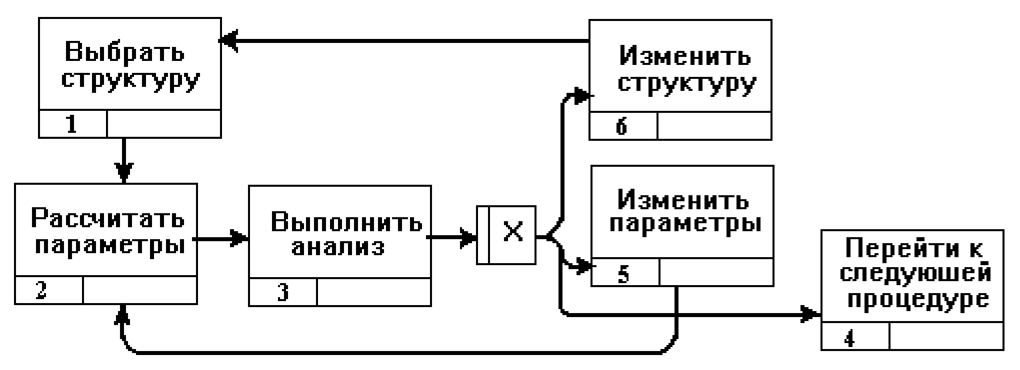


Рис. 6.5. IDEF3-диаграмма последовательсноти операций

тельность выполнения функций, изображаются сплошными линиями-стрелками. Для указания разветвлений и слияний связей (их принято называть перекрестками) используют квадраты, у которых одна или обе вертикальные стороны представлены двойными линиями, а внутри квадрата записан один из символов &, О или X. При разветвлении эти символы означают реакцию всех, некоторых или только одной из последующих функций на входное воздействие соответственно. Аналогичный смысл имеют символы &, О или X при слиянии – последующая функция начинает выполняться после окончания всех, некоторых или только одной из входных операций.

На рис. 6.6 представлены пример объект-ориентированной IDEF3-диаграммы. В таких диаграммах имеются средства для изображения состояний системы, активностей, переходов из состояния в состояние и условий перехода.

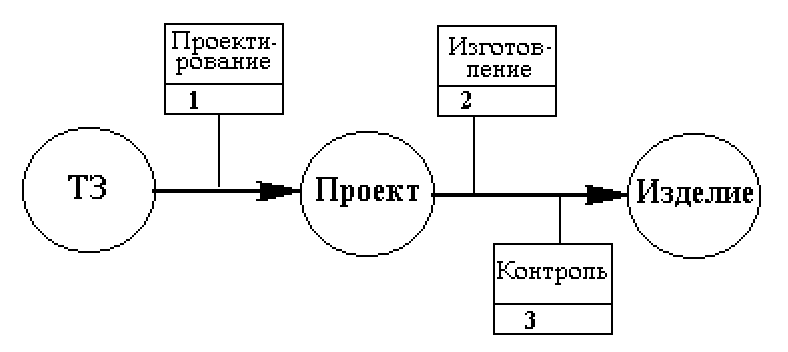


Рис. 6.6. IDEF3- диаграмма перехода состояний

Диаграммы IDEF0 или IDEF3 могут быть преобразованы в имитационные модели, если задать дополнительные свойства функций, характеризующие затраты ресурсов. Чаще всего имитационные модели представляют в виде сетей Петри. Преобразование связано с введением времени в функциональную IDEF0 или в поведенческую IDEF3-модель, с заменой функций переходами, а объектов, отождествляемых со стрелками блоков ICOM, с метками в сетях Петри.

Методика IDEF1X. IDEF1 — методика *информационного (инфологического)* проектирования приложений, в настоящее время применяется ее усовершенствованный вариант IDEF1X. В IDEF1X имеется ясный графический язык для описания объектов и отношений в приложениях. Это язык диаграмм сущность-связь.

Основные компоненты описаний в IDEF1X: сущности (блоки), отношения (связи), атрибуты.

Сущность — множество объектов, обладающих общими свойствами (в языках программирования понятие сущности совпадает с понятием типа). Конкретные элементы этого множества называют *экземплярами* сущности. Атрибуты характеризуют свойства сущностей, их значения однозначно идентифицируют экземпляры сущностей. Если сущность **А** может быть определена только с помощью ссылки на свойства некоторой другой сущности **B**, то **А** называется зависимой (дочерней) сущностью, а **В** выступает в роли родительской сущности.

Сущности в IDEF1X-диаграммах изображаются в виде прямоугольников, при этом у зависимых сущностей углы прямоугольников должны быть скругленными.

Отношения (связи) между сущностями в IDEF1X являются бинарными отношениями. Выделяют *идентифицирующие* отношения — связи типа родитель-потомок, в которых потомок (зависимая сущность) однозначно определяется своей связью с родителем, и неидентифицирующие отношения, означающие, что у связанного этим отношением экземпляра одной сущности может быть, а может и не быть соответствующего экземпляра второй сущности (пример идентифицирующего отношения изготовитель-товар, неидентифицирующего отношения — рабочая станция — дигитайзер). Иденти-

фицирующее отношение изображают на диаграмме сплошной линией между прямоугольниками связанных сущностей, неидентифицирующее отношение показывают пунктирной линией. На дочернем конце линии должно быть утолщение (жирная точка). Мощность k связи – число экземпляров зависимой сущности, соответствующее одному экземпляру родительской сущности. Известное значение мощности может быть указано около утолщенного конца линии связи. При этом символ p означает $k \geq 1$, а символу z соответствует $k = 0$ или 1. Отсутствие символа интерпретируется $k \geq 0$.

Различают также специфические и неспецифические отношения. *Неспецифические* отношения — это связи типа “многие ко многим” и обозначаются сплошной линией с утолщениями на обоих концах.

В отношениях родитель-потомок возможно наличие у потомка единственного родителя (характеристическая связь) или нескольких родителей (ассоциативная связь). Выделяют также отношения категоризации (наследования), отражающие связи между некоторой общей сущностью и вариантами ее реализации (категориями). Примером категориальной связи является отношение тип прибора — альтернативные варианты этого прибора.

Среди атрибутов различают ключевые и неключевые. Значение *ключевого атрибута (ключа)* однозначно идентифицирует экземпляр сущности. *Внешний ключ* – это атрибут (или атрибуты), входящий в ключ родителя и наследуемый потомком. На IDEF1X-диаграммах ключи записывают в верхней части прямоугольника сущности, причем внешние ключи помечают меткой FK (Foreign Key), неключевые атрибуты помещают в нижнюю часть прямоугольников. В идентифицирующих отношениях все ключи родителя входят и в ключи потомка, в неидентифицирующих ключи родителя относятся к неключевым атрибутам потомка.

Нормальные формы отношений позволяют выявить атрибуты, которые целесообразно (с целью устранения избыточности) считать сущностями. Известно несколько нормальных форм, обычно используют первые три из них.

Первая нормальная форма требует, чтобы шапка таблицы (отношения) была одноэтажная (т.е. все атрибуты характеризуются атомарными значениями), строки-дубли должны быть устранены.

Вторая нормальная форма устанавливается для сущностей, удовлетворяющих условиям первой нормальной формы и имеющих составные ключи. Она определяется отсутствием атрибутов, зависящих только от части составного ключа. Подобные атрибуты должны быть выделены в отдельные сущности.

Третья нормальная форма дополнительно характеризуется отсутствием транзитивных связей (взаимозависимости) атрибутов.

Разработка информационной модели по IDEF1X выполняется за несколько стадий.

<u>Стадия 0.</u> Выяснение цели проекта, составление плана сбора информации. Обычно отправным пунктом для разработки информационной модели является IDEF0-модель.

<u>Стадия 1.</u> Выявление и определение сущностей. Это неформальная процедура.

<u>Стадия 2.</u> Выявление и определение основных отношений. Результат представляется или графически в виде ER-диаграмм или в виде матрицы отношений, элемент которой A_{ij}=1, если имеется связь между сущностями i и j, иначе A_{ij}=0. Транзитивные связи не указываются.

<u>Стадия 3.</u> Детализация неспецифических отношений, определение ключевых атрибутов, уста-

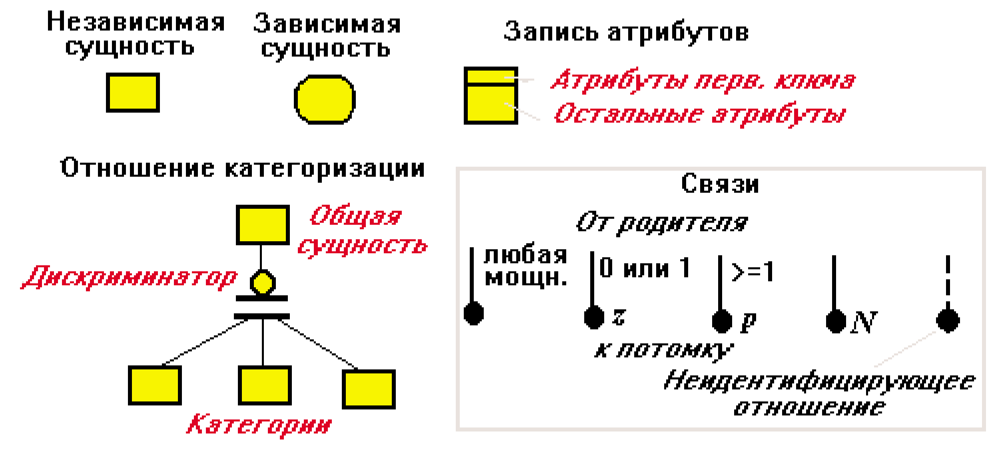


Рис. 6.7. Элементы языка IDEF1X

новление внешних ключей. Детализация неспецифических отношений заключается в замене связей "многие ко многим" ($M \leftrightarrow$ М) на связи "$M \leftrightarrow 1$" и "$1 \leftrightarrow M$" введением сущности-посредника. Например, отношение "преподаватель — студенческая группа" может быть заменено на отношения этих сущностей с сущностью-посредником "расписание".

Стадия 4. Определение атрибутов и их принадлежности сущностям.

Основные элементы графического языка IDEF1X представлены на рис. 6.7.

Между IDEF0 и IDEF1X-моделями одного и того же приложения существуют определенные связи. Так, стрелкам на IDEF0-диаграммах соответствуют атрибуты некоторых сущностей в IDEF1X-моделях, что нужно учитывать при построении информационных моделей.

Обзор других методик IDEF. Методика IDEF4 реализует *объектно-ориентированное проектирование* больших систем. При процедурном программировании кодированию предшествует удобное для пользователя изображение программы на графическом языке граф-схем или диаграмм потоков данных. Целесообразно иметь аналогичные средства, учитывающие специфику объектно-ориентированного программирования.

В частности, такие средства предоставляет IDEF4. Другим вариантом графического языка поддержки объектно-ориентированного проектирования ПО является язык UML (Unified Modeling Language), рассматриваемый консорциумом OMG на предмет стандартизации.

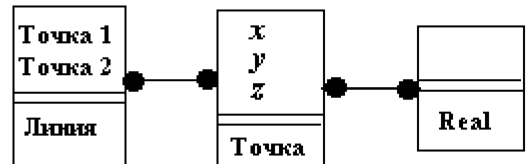


Рис. 6.8. IDEF4-диаграмма типов

Методика IDEF4 содержит графический язык для изображения взаимосвязей классов, атрибутов, методов в виде ряда диаграмм: типов, наследования, протоколов, клиентов, таксономии методов. Примеры диаграмм приведены на рисунках. В этих диаграммах прямоугольники с поперечными линиями соответствуют классам, имена которых указаны ниже поперечных линий, а сверху линий записаны идентификаторы атрибутов. Процедуры (методы) в IDEF4 изображены прямоугольниками без поперечных линий. Передаваемые параметры записаны в овальных фигурах.

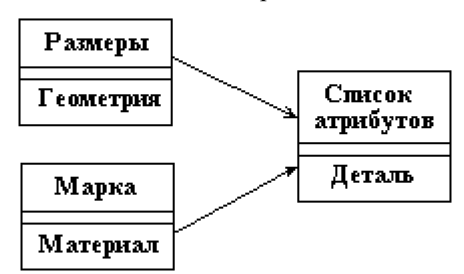


Рис. 6.9. IDEF4-диаграмма наследования

Примеры диаграмм типов данных и наследования приведены на рис. 6.8 и 6.9 соответственно. В примере рис. 6.9 объекты класса "Деталь" наследуют часть атрибутов из классов "Геометрия" и "Материал".

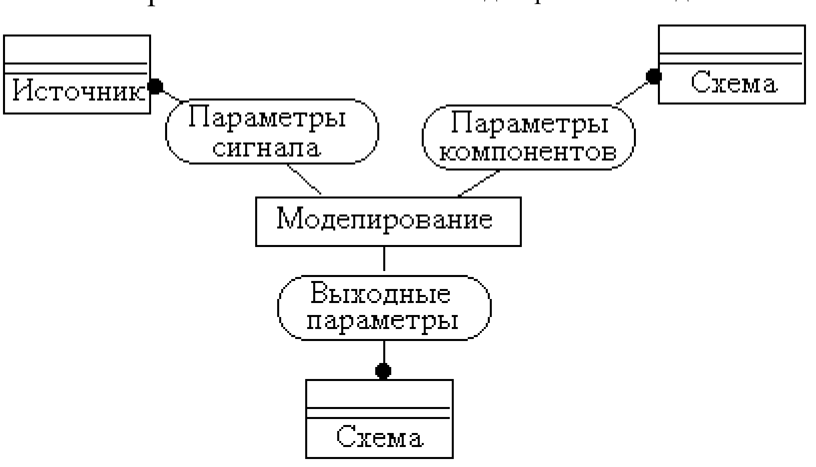


Рис. 6.10. IDEF4-диаграмма протоколов

Из рис. 6.10 ясно, что для процедуры моделирования некоторой схемы входными параметрами являются атрибуты источников сигналов и параметры компонентов схемы, а результатом — значения выходных параметров.

На рис. 6.11 показан пример классификации методов, согласно которой методы решения перечисленных частных задач относятся к методам дискретной оптимизации.

Связи вызывающих и вызываемой процедур представлены на рис 6.12.

Методика IDEF5 направлена на представление *онтологической информации* приложения в удобном для пользователя виде. Онтология связана с определениями и понятиями, используемыми для характеристики объек-

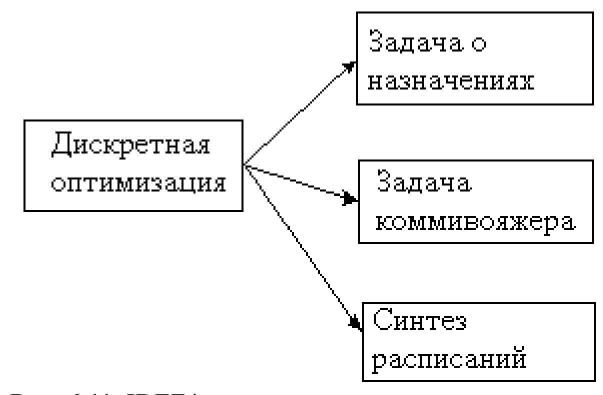


Рис. 6.11. IDEF4-диаграмма таксономии методов

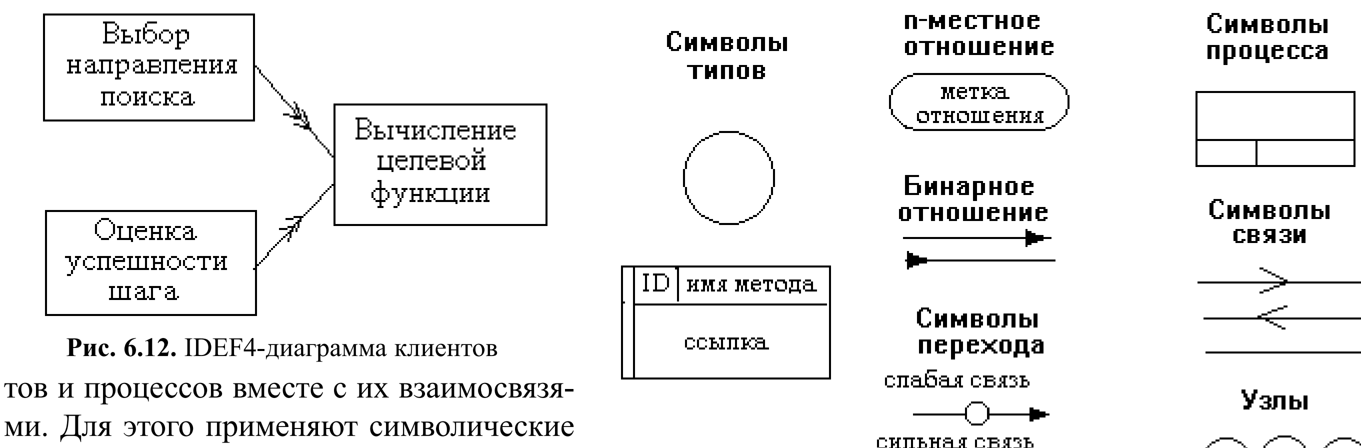


Рис. 6.12. IDEF4-диаграмма клиентов

Рис. 6.13. Символы графического языка IDEF5

тов и процессов вместе с их взаимосвязями. Для этого применяют символические обозначения (дескрипторы) объектов, их ассоциаций, ситуаций и схемный язык описания отношений (классификации, часть-целое, перехода и т.п.), составляют словарь дескрипторов. В методике имеются правила связывания объектов (термов) в правильные предложения, языковые механизмы для установления соответствия между объектами реального мира и их идентификаторами (дескрипторами).

В IDEF5 имеются две части: 1) схемный язык; 2) язык разработки (elaboration). Основные символы схемного языка представлены на рис. 6.13, пример классификационной схемы — на рис. 6.14 и пример диаграммы перехода состояний с символикой IDEF5 — на рис. 6.15.

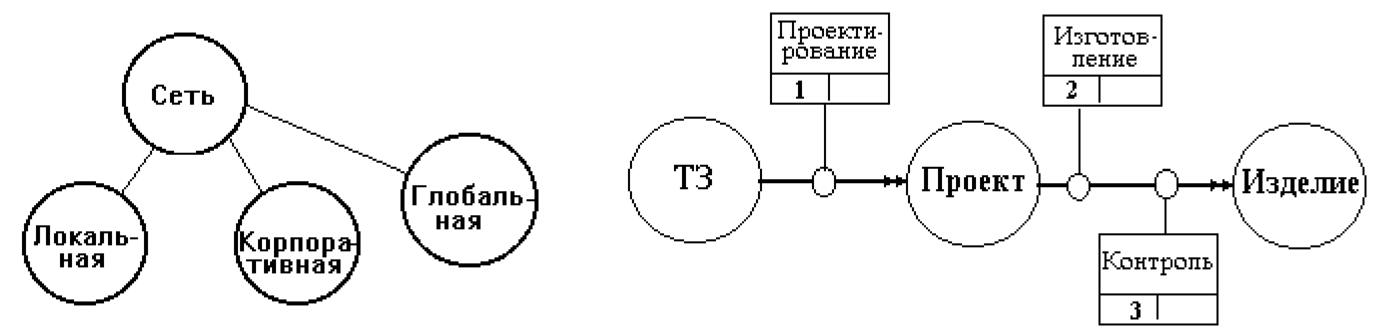


Рис. 6.14. Диаграмма классификации

Рис. 6.15. Диаграмма перехода состояний в IDEF5

Развитие методик реинжиниринга (BPR Business Process Reenginiring) продолжается в США по программе IICE (Information Integration for Concurrent Engineering). Разработаны, но пока (1998 г.) не получили официального статуса от органов стандартизации методики, имеющие индексы IDEF6, 8, 9, 14, разрабатываются методики IDEF7, 10, 12.

IDEF6 (Design Rationale Capture) направлена на получение и представление решений по выбору стратегии проектирования и обоснованию предпринятых шагов. В отличие от других методик IDEF, в которых фиксируются результаты проектирования, в IDEF6 главный упор сделан на пути получения этих результатов и обоснование промежуточных решений. Такой подход особенно важен при разработке сложных систем в недостаточно определенных ситуациях. Фиксация шагов и обоснований помогает при дальнейших модернизациях систем, сохранению и использованию рационального опыта проектирования. Методика упорядочивает обнаружение и устранение неопределенностей, ошибок, неудовлетворенных ограничений. Язык методики включает предложения, связывающие компоненты проекта с пунктами обоснования. Под компонентами проекта обычно подразумевают компоненты, отражаемые на диаграммах IDEF0-5, например, стрелки ICOM из IDEF0, сущности, атрибуты, отношения из IDEF1X, объекты, сообщения, события из IDEF4 и т.п. В качестве пунктов обоснования могут фигурировать стандарты, экспериментальные данные, ограничения и т.п.

IDEF8 (Human-System Interaction Design) предназначена для проектирования взаимодействия человека с технической системой. Эта методика не является методикой создания графического пользовательского интерфейса и потому обычно дополняется некоторой системой GUI (Graphic User Interface). Здесь определяется содержание (на абстрактном уровне) той части работы, которую выполняет человек. Создаваемые сценарии должны удовлетворять ряду оговоренных в методике принципов таких, как уменьшение нагрузки на человека, идентичность средств диалога в разных системах, наличие обратной связи для исправления ошибок, хранение истории диалога, помощь советами по выполнению действий и т.д.

IDEF9 (Business Constraint Discovery) нацелена на выявление разнообразных ограничений (технических, физических, юридических, политических, организационных), которые должны быть учтены при разработке системы, и для анализа их влияния на принимаемые решения в процессе реинжиниринга. Обычно в качестве систем фигурируют сложные информационные системы с ориентацией на экономические и управленческие приложения. Ограничение — это отношение,

которое должно соблюдаться. Ограничения делятся на контексты (группы родственных ограничений). Применение IDEF9 заключается в выполнении нескольких шагов: 1) сбор свидетельств (фактов, указывающих на наличие ограничения); 2) классификация — определение контекстов, объектов, отношений; 3) прогнозирование — выявление ограничений на основе свидетельств; 4) отбор значимых ограничений; 5) определение экспертов для тестирования результатов; 6) детализация и фильтрация ограничений. В методике даны рекомендации по выполнению этих шагов. Предлагается графический язык, элементами которого являются система, блоки ограничений, контексты, линии связи, логические связки OR, AND, XOR (исключающее ИЛИ).

IDEF14 (Network Design) предназначена для проектирования корпоративных вычислительных сетей, их представления на графическом языке с описанием конфигураций, очередей, сетевых компонентов, требований к надежности и т.п. Чаще всего методика применяется для модернизации уже существующих сетей. Поэтому в ней предусматривается разработка моделей как “AS IS”, так и “TO BE”. Проектирование включает в себя определение топологии сети или схемы коммуникаций, реализацию нужного качества обслуживания, анализ функционирования (трафик, дисциплины обслуживания в узлах, протоколы доступа). Модель топологии дополняется моделями очередей, надежности, материальных затрат. Важную роль играет библиотека методов построения и компонентов сетей. Методика основана на выполнении ряда шагов: установление целей модернизации, исследование существующей сети, определение типов компонентов в ней, построение модели “AS IS”, ее верификация, анализ результатов, корректировка с переходом к “TO BE”. В графическом языке IDEF14 сети и подсети изображаются в виде облаков, топологические связи представляются линиями, для узлов используются специальные иконки, возможны поясняющие надписи, список характеристик размещается в прямоугольниках.

Унифицированный язык моделирования UML. Язык UML положен в основу Rational Unified Process (RUP) — известной методологии проектирования информационных систем, развиваемой фирмой Rational Software. В UML также используется ряд диаграмм.

К основным следует отнести прежде всего диаграммы классов. Они имеют следующие отличия от аналогичных диаграмм в IDEF4.

Во-первых, в прямоугольнике класса имеются три секции, в верхней секции записывается имя класса, в средней секции — атрибуты, в нижней части — процедуры класса. При записи атрибутов указываются символ доступности (+ — public, # — protected, - - private), идентификатор атрибута, тип атрибута. Запись процедуры аналогична подобным записям в языках программирования: указывается имя процедуры и в скобках — список параметров.

Во-вторых, в диаграммах классов UML отображение отношений часть-целое (отношений агрегации) выполняется с помощью линий с ромбовидной стрелкой, направленной от класса-части к классу-целому, и отношений наследования (суперкласс-подкласс) — с помощью линий с обычной стрелкой, направленной от подкласса к суперклассу.

Поведенческий аспект моделирования отражен в диаграммах процессов, имеющихся в UML. Они бывают двух типов — диаграммы сценариев (ДС) и диаграммы взаимодействия объектов (ДВО).

Сценарий — это последовательность событий, заключающихся в воздействиях (посылках сообщений) одного объекта на некоторый другой объект. В ДС объекты изображаются прямоугольниками и располагаются в горизонтальном ряду объектов. Ось времени направлена от этого ряда вертикально вниз. От каждого объекта параллельно оси времени идут так называемые их линии жизни (lifelines). Каждое событие изображается горизонтальной линией со стрелкой от линии жизни объекта, посылающего сообщение, к линии жизни объекта, принимающего сообщение. Над этими линиями возможен поясняющий текст. Линии располагаются одна над другой в порядке, в котором события совершаются (пример ДС на рис. 6.16).

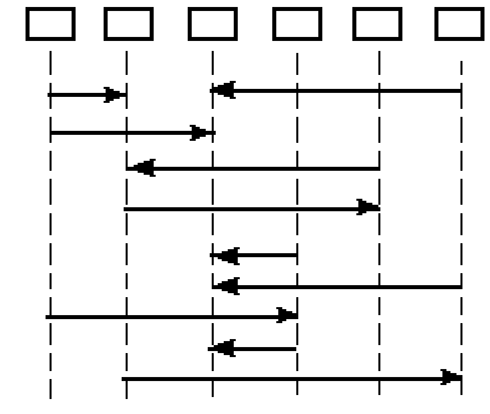


Рис. 6.16. Вид диаграммы сценариев

Диаграмма ДВО представляет собой граф, в котором вершины соответствуют объектам, а ребра — воздействиям. Около ребер возможны поясняющие записи, в частности, последовательные номера, указывающие порядок совершения событий.

К числу других диаграмм относятся диаграммы использования, цель которых — отобразить взаимодействие системы с пользователем. В этих диаграммах отображены в виде овалов те функции, которые непосредственно должен (или может) выполнять пользователь. При этом пользователи различаются ролями, выполняемыми ими при эксплуатации системы.

Проектирование информационной системы в RUP начинается с построения диаграмм использования. При этом определяется и согласовывается внешняя функциональность системы и в итоге фор-

мируется техническое задание на разработку ПО. Далее разрабатываются диаграммы взаимодействия “пользователь-система”, при этом выявляются необходимые объекты, строятся диаграммы классов, формируется компонентная структура ПО.

Программное обеспечение CASE-систем для концептуального проектирования. На рынке программных продуктов имеется много CASE-систем для концептуального проектирования АС.

Чаще всего в них поддерживается методология IDEF. В России широко известны программы BPwin, ERwin, OOwin фирмы Platinum Technology, Design/IDEF фирмы Meta Software, CASE-Аналитик фирмы Эйтэкс, Silverrun фирмы CSA и др.

BPwin (Business Processing) предназначена для разработки функциональных моделей по методике IDEF0.

ERwin предназначена для разработки информационных моделей по методике IDEF1X. Имеются средства, обеспечивающие интерфейс с серверами БД (от пользователя скрыто общение на SQL-языке), перевод графических изображений ER-диаграмм в SQL-формы или в форматы других популярных СУБД. Предусмотрены интерактивные процедуры для связывания дуг IDEF0 с сущностями и атрибутами IDEF1X, т.е. для установления связей между BPwin и ERwin.. В систему включены также типичные для CASE средства разработки экранных форм.

OOwin служит для поддержки объектно-ориентированных технологий проектирования информационных систем. Один из способов использования OOwin — детализация объектно-ориентированной модели на базе созданной ER-модели. При преобразовании ER в OO-представление сущности и атрибуты становятся классами (множествами подобных объектов). Классы могут быть дополнены описанием услуг класса, т.е. выполняемых операций, передаваемых и возвращаемых параметров, событий. Другой способ использования OOwin — реинжиниринг, так как модернизация проводится на уровне существующей модели.

Система Design/IDEF (фирма Meta Software) предназначена для концептуального проектирования сложных систем. С ее помощью разрабатываются спецификации, IDEF0 и IDEF1X-диаграммы, словари данных, проводится документирование и проверяется непротиворечивость проектов. Имеется дополнительная система Design/CPN, позволяющая проводить имитационное моделирование на основе моделей, преобразованных в цветные сети Петри.

Другой известной инструментальной средой моделирования приложений является Designer/2000 фирмы Oracle. Модель приложения может быть сгенерирована по ответам пользователя на вопросы системы. Используются собственные методики Oracle, позволяющие строить диаграммы потоков данных, сущность-отношение, иерархические деревья данных с возможностью их представления в SQL формах и, следовательно, поддерживается связь с любыми СУБД, работающими в ODBC.

Система Silverrun (фирма Computer Systems Advisors) предназначена для анализа и проектирования информационных систем. Реализовано раздельное функциональное и информационное моделирование. Включает в себя четыре основные подсистемы: моделирование бизнес-процессов, построение моделей сущность-отношение, инфологическое проектирование реляционных баз данных, управление групповой работой. Имеет интерфейс к Oracle, Informix, Sybase и ряду других СУБД.

Среди отечественных систем выделяется CASE Аналитик, в которой выполняется построение диаграмм потоков данных, получение отчетов, генерация макетов документов и др. Имеется интерфейс к ERwin.

Методология объектно-ориентированного анализа и проектирования ПО по методике Г.Буча с использованием языка UML реализована в системах Rational Rose (фирма Rational Software Corporation) и Platinum Paradigm Plus (фирма Platinum Technology). В Rational Rose поддерживается генерация кода по построенным диаграммам классов, обратное моделирование (т.е. построение UML-модели по программному коду на таких языках, как C++, Java, Visual Basic, IDL CORBA), визуальное программирование. Язык UML применяют и в ряде других систем, например, в инструментальной среде объектно-ориентированного проектирования ПО objectiF (фирма micro TOOL), в которой автоматически генерируется программный код по графическому UML-описанию.

Ряд программных продуктов, реализующих IDEF-модели, разработаны фирмой KBSI, в частности, ProSim реализует IDEF3, SmartER — IDEF1 и IDEF1X, SmartClass — IDEF4.

Поведенческое моделирование предприятий предусмотрено также в некоторых системах реинжиниринга, например, в системе BAAN IV.

Для преобразования функциональных или поведенческих моделей в имитационные применяют специальные программы. Так, вместе с программой BPWin для получения имитационных моделей используют программу BPSimulator. Преобразование IDEF0-модель → сеть Петри реализовано в таких программах, как CPN/Design (фирма Meta Software) со специальным языком программирования ML, ProTem (Software Consultants International Limited) с вариацией типов меток, PACE (Grossenbacher software) с программированием на языке Smalltalk.

Метамодели и стандарты CDIF (CASE Data Interchange Format). Метамодель — средство, являющееся инвариантным к частным представлениям индивидуальных пользователей, служащее промежуточным звеном в процедурах взаимодействия приложений, характеризуемых своими локальными моделями.

Место метамодели в информационных процессах взаимодействия иллюстрирует рис. 6.17. Из рисунка ясно, что вместо непосредственного обращения одного приложения к другому, при котором каждое приложение должно иметь конверторы всех других локальных моделей, используется транс-

ляция передаваемой информации на промежуточный язык метамодели, а принимающее приложение переводит метамодельное представление в свой собственный формат. Метамодельный подход имеет ряд преимуществ, например, каждое приложение становится открытым и может развиваться независимо от других, система не имеет ограничений на включение новых приложений.

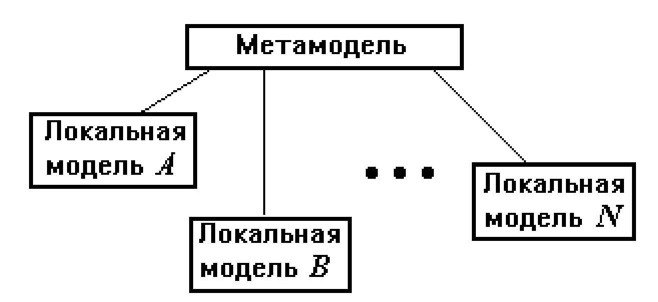


Рис. 6.17. Место метамодели в процессах информационного обмена.

Примерами метамоделей могут служить технология ODBC взаимодействия различных СУБД, основанная на языке SQL, графические системы типа GKS, концепция байт-кодов в языке Java и т.п.

В технологиях проектирования АС и реинжиниринга предприятий важное место отводится разработке метамоделей, направленных на взаимную трансформацию функциональных, информационных и структурных моделей. Для этого, в частности, требуется систематизация понятий, фигурирующих в приложениях, и построение словарей соответствия моделей этих типов.

Другое важное назначение метамоделей — интеграция CASE-средств разных производителей. Такая интеграция требуется, например, при недостаточных возможностях каждого из доступных CASE пакетов в отдельности, для доступа в условиях изменения программного и лингвистического обеспечений к информации, разработанной с помощью разных версий CASE-систем и накапливающейся длительное время в архивах.

Целям интеграции CASE-средств разных производителей служат стандарты серии CDIF, разрабатываемые организацией **EIA** (Electronics Industries Association) и признаваемые Международной организацией стандартизации **ISO** (International Standard Organization).

Метамодель в CDIF определяется, как средство, с помощью которого осуществляется правильная интерпретация данных при их передаче из одной CASE-среды в другую. Такая интерпретация требуется при взаимодействии сред, использующих различные формы представления однородной в смысловом отношении информации. Другими словами, метамодель применяют для передачи и правильной интерпретации данных с одинаковой семантикой, но с разным представлением в частных CASE системах. Например, данные, близкие в семантическом отношении, но различающиеся по представлению, фигурируют в методиках информационного моделирования (data modeling), моделирования потоков данных (data flow modeling), событийного моделирования переходов состояний (state event modeling), объектно-ориентированного анализа и проектирования (object oriented analysis and design). CDIF-метамодель осуществляет интерфейс между ними.

Программное обеспечение, поддерживающее CDIF, позволяет представлять данные в желаемой форме (в соответствии с предметной областью). Например, конечно-автоматная модель может быть представлена в форме графа или матрицы перехода состояний, объектно-ориентированная модель — с использованием прямоугольников или произвольно очерченных фигур и т.п. Клиент, поддерживающий CDIF, транслирует форму источника информации в форму, доступную клиенту с сохранением семантики данных.

Очевидно, что для каждой предметной области, характеризуемой своим множеством семантически близких понятий можно построить свою метамодель. Такие предметные области в стандартах CDIF называют Subject Areas, для многих предметных областей разработаны свои CDIF-стандарты (метамодели). Очевидно также, что потребности в метамоделях могут возникать для новых предметных областей, поэтому в CDIF отдельная методика посвящена включению в стандарты новых метамоделей. Имеются также общие для различных предметных областей компоненты метамоделей. Обычно интегрированная метамодель строится на основе парадигмы сущность-отношение.

Обменный файл в CDIF состоит из трех частей: заголовка (имя, дата, источник, способ кодирования и другие общие атрибуты), метамодели (указывается тип используемой метамодели) и собственно передаваемых данных.

Список стандартов CDIF приведен в приложении. Стандарты подразделены на три группы. Первая группа содержит обзор стандартов CDIF и общие правила их расширения.

Вторая группа определяет форматы представления данных, т.е. синтаксис и способы кодирования передаваемых данных.

Третья группа содержит стандарты, ориентированные на представление семантики передаваемых данных. Каждый из стандартов относится к определенной предметной области. Например, есть стандарты или проекты стандартов для таких областей, как объектно-ориентированный анализ и проектирование, моделирование бизнес-процессов, проектирование автоматизированных систем управления, описание потоков данных, данных в реляционных базах данных и др. Кроме того, введены иерархическая структура метамодели и возможности наследования, благодаря выделению наиболее общих частей, справедливых для многих предметных областей, и их представлению в отдельных стандартах.

Таким образом, в метамодели CDIF имеет место отделение семантики от способа представления данных. Правильная передача семантики сочетается с варьированием форм представления данных.

6.3. STEP-технология

Общие сведения о стандартах сопровождения промышленной продукции на всех этапах ее *жизненного цикла.* STEP (Standard for Exchange of Product data) — это совокупность стандартов (под номером ISO 10303), определяющих средства описания (моделирования) промышленных изделий на всех стадиях жизненного цикла. Совокупность стандартов STEP лежит в основе CALS-технологий.

Единообразная форма описаний данных о промышленной продукции обеспечивается введением в STEP языка Express, инвариантного к приложениям. Стандарты STEP не отрицают, а развивают методику информационного моделирования IDEF1X и предполагают возможность совместного использования с методикой функционального проектирования IDEF0 и рядом других международных стандартов (например, со стандартами ISO P-LIB, Mandate, SGML, CDIF и стандартом EIA 649).

Стандарт ISO 10303 состоит из ряда документов (томов).

Том ISO 10303-1 — вводный стандарт, описывающий структуру всей совокупности томов и основные принципы STEP. В следующих группах томов содержатся описания инвариантного к приложениям языка Express, даны методы его реализации, модели, ресурсы, как общие для приложений, так и некоторые специальные (например, геометрические и топологические модели, описание материалов, процедуры черчения, конечно-элементного анализа и т.п.), прикладные протоколы, отражающие специфику моделей в конкретных предметных областях, методы тестирования моделей и объектов.

Удовлетворению требований создания открытых систем в STEP уделяется основное внимание — специальный раздел посвящен правилам написания файлов обмена данными между разными системами, созданными в рамках STEP-технологии.

Развитие линии стандартов STEP находит выражение в разработке новых стандартов Parts Library (ISO 13584), Parametrics (ISO 14959), Mandate (ISO 15531).

Стандарты Parts Library (P-LIB) содержат обзор и основные принципы представления данных о стандартных компонентах промышленных изделий. В этих стандартах представлены в виде библиотек данные о семействах таких типовых широко используемых компонентов изделий, как болты, подшипники, электронные компоненты и т.п., с целью использования этих данных в системах автоматизированного проектирования. В P-LIB содержатся также правила использования, интерфейса и модификации библиотечных описаний. Цель стандарта — обеспечить инвариантный для приложений механизм оперирования частями библиотеки.

Благодаря ISO 13584 различные прикладные САПР могут разделять данные из обобщенных баз, беспрепятственно обмениваться данными о типовых компонентах.

Стандарты P-LIB состоят из нескольких частей.

Часть 1 — обзор и основные положения серии стандартов.

Номера с 10 по 19 отведены для частей, содержащих концептуальные положения.

Номера с 20 по 29 выделены для описания логических ресурсов. Здесь разработаны части: 20 — общие ресурсы; 24 — логическая модель поставляемой библиотеки (Logical model of supplier library); 26 — определение поставщиков (Supplier Identification).

Номера 30-39 используются для описания ресурсов внедрения. Здесь разработана часть 31 — интерфейс геометрического программирования (Geometric Programming Interface).

Описание методологии структуризации семейств содержится в части 42.

Протоколам обмена посвящены части, начинающиеся с номера 101. Часть под номером 101 содержит протокол обмена геометрической параметризованной информацией; часть под номером 102 — протокол обмена согласованными с STEP данными.

Стандарты Parametrics введены сравнительно недавно (1996 г.) в связи с тем, что стандарты STEP в недостаточной мере учитывали особенности современных САПР в части представления параметризованных моделей изделий и обмена параметризованными данными.

Рабочая группа ISO по Parametrics решает как краткосрочные, так и перспективные задачи. Первые из них касаются удовлетворения потребностей геометрического проектирования и машинной графики в сегодняшних САПР, в которых широко используются параметризованные модели. Вторые касаются попыток распространения идей параметризации на более ранние этапы проектирования и на более широкий круг моделей и процедур проектирования, имеющих не только геометрический характер.

Стандарты Mandate посвящены представлению данных, относящихся к функционированию предприятий, управлению территориально распределенными производственными системами, обмену данными о производстве с внешней для предприятия средой.

Часть стандарта, обозначаемая ISO 15531-21, содержит обзор и основные принципы представления данных о промышленной продукции. Содержание этой части характеризуется следующими ключевыми словами: системы промышленной автоматизации и интеграция, промышленные данные, обмен данными об управлении производством, обмен данными с внешней средой.

Том ISO 15531-31 посвящен обзору и основным принципам использования данных о производственных ресурсах. Излагаются модель, форма и атрибуты представления данных о производственных ресурсах, об управлении их использованием.

Том ISO 15531-41 содержит обзор и основные принципы управления потоками производственных данных.

Семейство стандартов SGML (ISO 8879) предназначено для унификации представления текстовой информации в АС. В цикле проектирования промышленной продукции стандарты SGML обслуживают стадию, на которой выполняется документирование результатов. Стандартная форма документов способствует их правильной передаче, интерпретации и многократному использованию многими системами и пользователями. Стандарты SGML разрабатывались прежде всего применительно к текстовым документам, но их возможности шире, так они применимы для документирования гипермедийных данных.

Роль стандартов SGML конкретизируется следующими направлениями их использования.

1. Единообразное представление структуры данных, включая классификацию и идентификацию типов документов, поддержку различных типов символов и языковых ограничений.

2. Дополнение моделей промышленных изделий, задаваемых в настоящее время стандартами STEP, моделями документов.

3. Обмен данными между различными АС, электронными или традиционными средствами публикации и прежде всего между STEP и SGML средами. Для достижения этой цели SGML-формы должны быть согласованны с формой обменного файла STEP, описываемого в томе ISO 10303-21.

Использование возможностей SGML в STEP-ресурсах осуществляется с помощью информационной структуры SGML_STRING, включаемой в модели на языке Express. Эта структура содержит информацию о требуемом документальном оформлении данных и, следовательно, позволяет выполнять в STEP-среде перечисленные выше функции SGML. Тем самым реализуется интеграция STEP- и SGML-стандартов.

Стандарт SGML устанавливает такие множества символов и правил для представления информации, которые позволяют правильно распознавать и идентифицировать эту информацию. Названные множества описывают в отдельной части документа, называемой декларацией DTD (Document Type Declaration), которую помещают в начале SGML-документа. В DTD указывают соответствие символов и их кодов, максимальные длины используемых идентификаторов, способ представления ограничителей для тегов (примером может служить символ “<” для тегов в HTML), другие возможные соглашения, синтаксис DTD, а также тип и версия документа. Следовательно, SGML можно назвать способом описания семейства конкретных языков разметки. В частности, подмножествами SGML явля-

ются языки разметки XML и HTML.

Стандарт EIA 649 посвящен *управлению конфигурацией* изделий. В нем установлены базовые принципы управления конфигурацией и правила управления внесением изменений в документацию, рассматриваются такие вопросы, как идентификация документа, взаимосвязи конфигурации продукта и данных, контроль версий данных и доступа к данным и др. В стандарте вводятся уровни статуса данных, к которым относится документ на том или ином этапе своего жизненного цикла. Возможны уровни рабочих, выпущенных, представленных и утвержденных данных. На уровне рабочих данных с документом работает его составитель (разработчик). На уровне выпущенных данных документ доступен соответствующим подразделениям организации-изготовителя, здесь и далее любое изменение данных требует выполнения специальных согласительных процедур. Представленные данные уже доступны для просмотра заказчикам (потребителям). Статус утвержденных данные получают после одобрения заказчиком.

Структура стандартов STEP. Построение открытых распределенных АС для проектирования и управления в промышленности составляет основу современной CALS-технологии. Главная проблема их построения — обеспечение единообразного описания и интерпретации данных, независимо от места и времени их получения в общей системе, имеющей масштабы вплоть до глобальных. Структура проектной, технологической и эксплуатационной документации, языки ее представления должны быть стандартизованными. Тогда становится реальной успешная работа над общим проектом разных коллективов, разделенных во времени и пространстве и использующих разные CAE/CAD/CAM-системы. Одна и та же проектная документация может быть использована многократно в разных проектах, а одна и та же технологическая документация — в разных производственных условиях, что существенно сократит и удешевит общий цикл проектирования и производства. Упрощается эксплуатация систем.

Эти цели поставлены при разработке стандартов STEP. К их разработке под эгидой ISO привлечен ряд ведущих специалистов фирм в разных отраслях промышленности.

Основу STEP составляет язык Express. Это язык унифицированного представления данных и обмена данными в компьютерных средах. Язык инвариантен к приложениям. Хотя он разрабатывался с ориентацией прежде всего на описание жизненных циклов промышленной продукции, области его применения значительно шире.

В STEP используются также следующие основные понятия:

AAM — *Application Activity Model*; это функциональная модель IDEF0 для определенного приложения;

ARM — *Application Requirements Model*; это модель данных, представленная обычными средствами приложения;

AIM — *Application Interpreted Model*; это ARM модель, переведенная в STEP представление;

AP — *Application Protocol*; это STEP стандарт, отражающий специфику конкретного приложения;

SDAI — *Standard Data Access Interface*; программный интерфейс к источникам данных (репозиториям) прикладных систем (в том числе к библиотекам моделей CAD/CAM систем) с переводом моделей в STEP файлы, используется в STEP средах для организации обменов между приложениями через общую базу данных STEP.

STEP — это совокупность стандартов и состоит из ряда томов. Тома имеют свои номера *N* и обозначаются как “часть *N*” или ISO 10303-*N*. Приведем краткую характеристику следующих основных групп томов:

— том ISO 10303-1 — вводный стандарт, выполняющий роль аннотации всей совокупности томов. В этом стандарте вводится ряд терминов, используемых в других стандартах, например, таких как продукт (product), приложение (application), проектные данные (product data), модель (model), модели AAM, AIM, ARM, прикладной протокол (AP), интегрированный ресурс (integrated resource), элемент функциональности (unit of functionality — UoF).

— части 11 - 14 — методы описания (Description methods),

— части 21 - 29 — методы реализации (Implementation methods),

— части 41 - 50 — интегрированные основные ресурсы (Integrated generic resources),

— части 101 - 108 — интегрированные прикладные ресурсы (Integrated application resources),

— части 201 - 236 — прикладные протоколы (Application protocols),

— части 501 - 520 — прикладные компоненты (Application interpreted constructs).

Списки избранных томов под номерами 31 - 35 – «Основы тестирования продукции» (Conformance testing methodology and framework) и 301 - 332 — «Абстрактные тестовые наборы» (Abstract test suites) приведены в приложении.

Методы описания. Первая группа документов — тома, с номерами в диапазоне с 11 до 19 отведены для описания диалектов языка Express.

N=11: *Express language reference manual*. Основное руководство по языку Express. Содержит также описания расширения Express-C базового языка и графического варианта языка Express-G. Базовый язык приспособлен для описания и передачи статических свойств объектов приложений, т.е. параметров структур и ограничений. Поэтому Express-C включает средства описания динамических свойств объектов (добавлено описание событий и транзакций). Для наглядности представления языковых конструкций в Express предусмотрены графические средства изображения моделей, в качестве которых может использоваться специальное дополнение Express-G (графический Express). Express-G — язык диаграмм, напоминающий язык описания информационных моделей в методике IDEF1X.

N=12: *Express-I Language Reference Manual*. Express-I — расширение языка, предназначенное для описания отдельных экземпляров данных.

Разрабатываются дополнения, относящиеся к следующим диалектам языка:

— Express-M: *Mapping definition languag*e; язык для описания соответствий между сущностями и атрибутами некоторых моделей, представленных в виде схем на языке Express. Например, этими схемами могут быть два разных прикладных протокола, имеющих частично общие данные, или две схемы одного приложения, но созданные разными лицами (при отсутствии соответствующего AP). Одна схема есть схема-источник, другая — целевая схема. Целевых схем может быть несколько при одной схеме-источнике. Предложения Express-M транслируются на язык C, результирующая программа представляет собой совокупность обращений к функциям базы данных SDAI в STEP-среде. Другими словами, транслятор относится к системе SDAI (см. протокол ISO10303-22), а Express-M можно рассматривать, как язык 4GL для обращений к функциям базы данных SDAI.

— Express-X: промежуточный язык, аналогичный Express-M и используемый для описания соответствий между типами данных в заданной исходной Express-схеме и создаваемыми новыми ее вариантами (views); в качестве views могут использоваться форматы с описанием того же множества сущностей, что и в Express-схеме, например, формат IGES (описанию языка Express-X посвящен будущий стандарт ISO 10303-14).

— Express-P: *Process definition language*; язык диаграмм для представления процессов, методов и коммуникационных структур.

— Express-V: язык, предназначенный для получения ARM представлений из AIM моделей, другими словами, для описания процедур поиска экземпляров Express-объектов, отвечающих заданным условиям, и доступа к ним, например, при создании новых ARM. Эти создаваемые ARM-представления обычно не требуют столь всестороннего описания приложения, как в AIM, и потому могут быть существенно проще. В Express-V имеются: 1) схема-источник (AIM), обычно это прикладной протокол, например, AP203; 2) схема-цель, задающая сущности, которые должны быть в создаваемой частной модели; 3) схема отображения нужных сущностей из источника в цель. На языке Express-V описываются условия (в виде клозов WHEN) такого отображения. берется подходящая уже существующая AIM, как источник, все совпадающие объекты переводятся в ARM, далее описываются оригинальные объекты. Дополнительной возможностью реализаций Express-V является обратное отображение специфики создаваемой ARM в исходную AIM с целью развития прикладных протоколов.

Для возможности применения языка Express должны быть разработаны методы реализации (Implementation Methods), которые могут быть представлены средствами файлового взаимодействия, построением БД, интерфейсом с языками программирования.

Методы реализации. Вторую группу (тома с номерами 21...29) называют “Методы реализации”, она служит для реализации межпрограммного информационного обмена между прикладными системами в STEP-среде. Предусмотрены межпрограммные связи с помощью обменного файла и доступа к БД.

N=21: *Clear Text Encoding of the Exchange Structure (physical transfer file format)*; стандарт устанавливает правила оформления обменного файла. Обменный файл играет в STEP важную роль; если собственно на языке Express определены сущности, то именно в обменном файле задаются экземпляры этих сущностей. Прикладные программы для связи со STEP средой должны читать и генерировать обменные файлы.

N=22: *Standard Data Access Interface Specification*; содержит описание SDAI — системы представления данных и доступа к данным конкретных прикладных систем (чаще всего это CAD/CAM системы). Данные, участвующие в межпрограммных связях, образуют SDAI-модели. В SDAI системе предусматривается компилятор кода, конвертирующего эти модели в SDAI базу данных, а также функции обращения к этой базе данных. Возможно непосредственное построение прикладных систем, работающих с SDAI базой данных.

Тома с номерами *N* = 23…29 устанавливают правила обращения к данным в SDAI базе данных на языках программирования C++, C*,* Java, на языке передачи данных в системах распределенных вычислений IDL, языке разметки XML.

Прикладные протоколы. *Прикладным протоколом* в STEP называют информационную модель определенного приложения, которая описывает с высокой степенью полноты множество сущностей, имеющихся в приложении, вместе с их атрибутами, и выражена средствами языка Express. Предполагается, что эта модель содержит в себе описание данных любой конкретной задачи соответствующего приложения, т.е. практические информационные модели прикладных задач оказываются частными случаями прикладных протоколов.

Прикладные протоколы в стандарте ISO 10303 содержатся в томах, начиная с *N*=201. Прикладные протоколы принято обозначать аббревиатурой AP с указанием номера, например, AP203, AP214. Для связи прикладной системы со STEP используемые ею данные должны быть описаны в соответствующем AP.

Список большинства разработанных прикладных протоколов приведен в приложении. Число прикладных протоколов в STEP может расширяться за счет разработки новых протоколов.

В прикладных протоколах широко используются типовые фрагменты информационных моделей, встречающиеся более чем в одном приложении. Эти фрагменты называют интегрированными общими и прикладными ресурсами.

Типовые фрагменты информационных моделей. Четвертая группа стандартов STEP (тома с номерами 41...50) “Интегрированные общие ресурсы” описывает общие для приложений ресурсы, под которыми понимаются основные компоненты (building blocks) для моделей прикладных протоколов. Например, описания геометрических объектов в виде поверхностей Безье или *B*-сплайнов могут использоваться во многих прикладных протоколах, поэтому эти описания вынесены в группу интегрированных ресурсов.

Тома с номерами 101 по 199 отведены для документов, относящихся к более специальным средствам, называемым интегрированными прикладными ресурсами (Integrated application resources).

Группа стандартов с номерами, начинающимися с *N* = 501, служит для описания данных о геометрических элементах и моделях некоторых конкретных типовых объектов и конструкций, часто используемых в ряде интегрированных ресурсов и прикладных протоколов.

Номера и названия томов, описывающих интегрированные и прикладные ресурсы и прикладные компоненты, приведены в приложении.

Организация в STEP информационных обменов. Возможны обмены через обменный файл и через БД SDAI. Эти способы поясняются на рис. 6.18 и 6.19 соответственно.

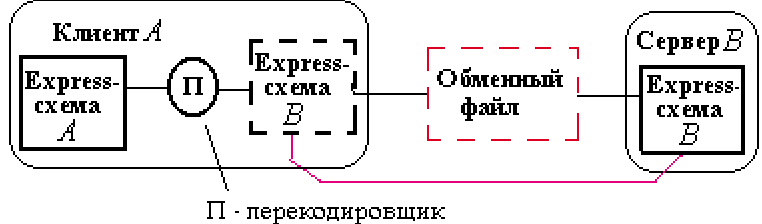


Рис. 6.18. Взаимодействие Express-приложений через обменный файл

Обменный файл (см. рис. 6.18) используется при связи двух систем *A* и *B*, имеющих общие данные с различными обозначениями. Пользователь должен написать перекодировщик (например, на языке Express-X), с помощью кото-

рого отождествляются идентификаторы одних и тех же сущностей, имевших разные обозначения в схемах *A* и *B*.

Связь через БД SDAI (рис.6.19) отличается от обмена по схеме рис. 6.18 тем, что здесь имеет место не просто обмен, а разделение данных многими пользователями, и SDAI фактически выступает в роли метамодели для разных САПР.

Описание языка Express в сокращенном виде приведено в следующем параграфе.

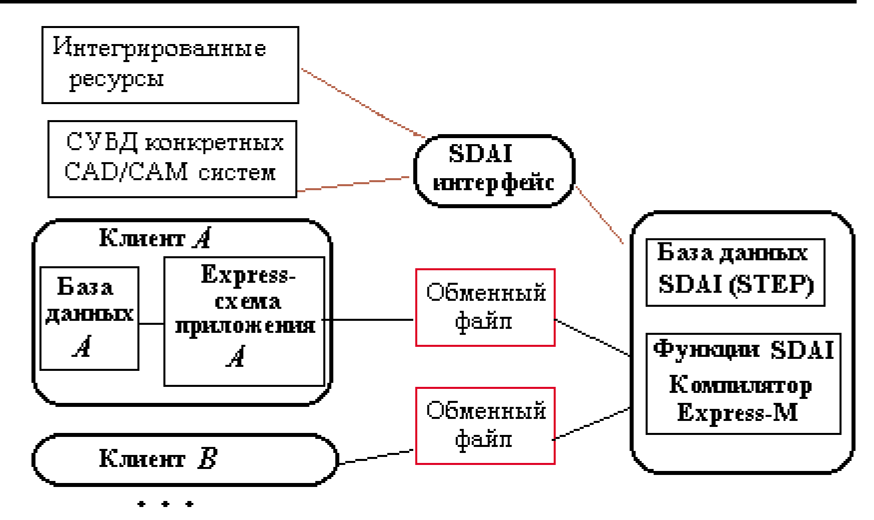


Рис. 6.19. Взаимодействие Express-приложений через базу данных SDAI

Стандарты управления качеством промышленной продукции. Международные стандарты серии ISO 9000 разработаны для управления качеством продукции, их дополняют стандарты серии ISO 14000, отражающие экологические требования к производству и промышленной продукции. Хотя эти стандарты непосредственно не связаны со стандартами STEP, их цели — совершенствование промышленного производства, повышение его эффективности — совпадают.

Очевидно, что управление качеством тесно связано с его контролем. Контроль качества традиционно основан на измерении показателей качества продукции на специальных технологических операциях контроля и выбраковкой негодных изделий. Однако есть и другой подход к управлению качеством, основанный на контроле качественных показателей не самих изделий, а проектных процедур и технологических процессов, используемых при создании этих изделий. Такой подход более эффективен и требует меньше затрат. Именно он и положен в основу стандартов ISO 9000.

Под *качеством* продукции в ISO 9000 подразумевается своевременное удовлетворение требований заказчика при приемлемой цене. Документальную систему с руководствами и описаниями процедур достижения качества называют *системой качества* (QS — Quality System).

Система качества обычно представляет собой совокупность трех слоев документов. Слои содержат: 1) описание политики управления для каждого системного элемента (организация, ответственные, контроль); 2) описание процедур управления качеством (что, где, кем и когда должно быть сделано); 3) тесты, планы, инструкции и т.п.

Сертификация предприятий по стандартам ISO 9001-9003 выполняется некоторой уполномоченной внешней организацией. Наличие сертификата качества — одно из важных условий для успеха коммерческой деятельности предприятий.

Стандарты ISO 14000 посвящены проблеме выполнения промышленными предприятиями экологических требований. По своей целевой направленности они близки стандартам управления качеством промышленной продукции ISO 9000, которые служат базой для ISO 14000. Стандарты ISO 14000 являются также системой управления влиянием на окружающую среду, они, как и ISO 9000, реализуются в процессе сертификации предприятий, задают процедуры управления и контроль документации, аудит, подразумевают соответствующее обучение и сбор статистики. Кроме требований заказчиков и покупателей, здесь воплощаются внутренние требования организации.

Краткое описание стандартов серий ISO 9000 и ISO 14000 дано в приложении.

6.4. Краткое описание языка Express

Структура описания приложения на языке Express. Базовый язык Express является объектно-ориентированным, имеет универсальный характер, его можно использовать для описания статических структур и их свойств в различных предметных областях, несмотря на то, что язык разрабатывался прежде всего в качестве средства представления моделей промышленных изделий на разных этапах их жизненного цикла.

Описание некоторого приложения на языке Express в рамках стандарта STEP называют *моделью (model)*. В модели декларируются множества понятий и объектов, входящих в приложение, свойства и взаимосвязи объектов.

Модель состоит из одной или нескольких частей, называемых *схемами (schema)*. Схема — раздел описания, являю-

щийся областью определения данных. В ней вводятся необходимые типы данных. При описании свойств типов данных могут применяться средства процедурного описания — процедуры, функции, правила, константы.

Схема. Описание схемы начинается с заголовка, состоящего из служебного слова **schema** и идентификатора — имени схемы. Далее следует содержательная часть — тело схемы. Описание заканчивается служебным словом **end_schema** (в этом и последующих примерах служебные слова языка Express выделены полужирным шрифтом)**:**

```
schema <имя схемы>;
<тело схемы>;
end_schema;
```



Рис. 6.20. Изображение схемы в Express-G

В языке Express-G схема представляется прямоугольником с разделительной горизонтальной линией, над этой линией записывается имя схемы, как это показано на рис. 6.20.

В теле схемы декларируются *типы данных (Data Type).* Тип данных — множество значений некоторой величины или множество объектов (набор экземпляров). В языке Express используются следующие типы данных: сущность *(Entity),* простой (*Simple Type*), агрегативный (*Aggregation Data Type*), определяемый (*Defined Data Type*), нечисловой (*Enumeration Data Type*) и выделяемый (*Select Data Type*) типы.

Сущности и атрибуты. *Сущность* — тип данных, представляющий набор концептуальных или реальных физических объектов с некоторыми общими свойствами, и служит для описания объектов предметной области. Свойства сущности выражаются в виде *атрибутов (Attributes).* К характеристикам сущностей относятся также ограничения, накладываемые на значения атрибутов или на отношения между атрибутами.

Описание сущности начинается со служебного слова **entity**, за которым следует идентификатор сущности, описание ее атрибутов и возможно также правил, каждый из атрибутов представлен его идентификатором и типом:

```
entity <имя сущности>;
<идентификатор атрибута>:<тип атрибута>;
. . .
end_entity;
```

Например, задание прямой линии (line) в виде двух инцидентных точек p0 и p1 (атрибутов типа point) выглядит следующим образом:

```
entity line;
p0,p1: point;
end_entity;
```

Атрибуты и переменные сами могут быть сущностями, так тип атрибутов предыдущего примера декларируется, как сущность, атрибутами которой в случае пространства 3*D* являются геометрические координаты x,y,z:

```
entity point;
x,y,z: real;
end_entity;
```

В Express-G сущности изображаются прямоугольниками, внутри прямоугольника записывается имя сущности (рис. 6.21).

Если свойство является необязательным для данной сущности, то его выражают так называемым *необязательным (optional) атрибутом*. В его описании перед типом атрибута добавляется служебное слово **optional**

```
<идентификатор атрибута>: optional <тип атрибута>;
```

<имя сущности>

Рис. 6.21. Изображение сущности в Express-G

Изображение атрибутов в Express-G поясняет рис. 6.22, из которого, в частности, ясно, что атрибут представлен прямоугольником, а связи “сущность-атрибут” или “сущность-сущность” отображаются линиями, причем в случае связи с **optional** атрибутом используется пунктирная линия. Направление связи обозначается окружностью на конце линии, ведущей к атрибуту. Имя атрибута записывается рядом с этой линией. В прямоугольнике атрибута записывается тип атрибута.

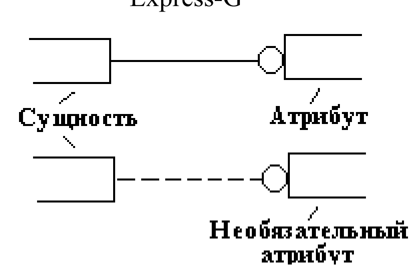


Рис. 6.22. Атрибуты в Express-G

Некоторые из атрибутов могут определяться через другие атрибуты. Тогда атрибуты, выражаемые через другие атрибуты, называют *порожденными (derived)*, что отображается служебным словом **derive** в декларации атрибута. Например, описание окружности, кроме обязательных атрибутов, которыми в нижеследующем примере выбраны радиус и центр окружности, может включать порожденный атрибут площадь круга:

```
entity point;
x,y,z: real;
end_entity;
entity circle;
center: point;
radius: real;                    -- явные атрибуты center, radius
derive
area: real := pi*radius**2;      (*порожденный атрибут area*)
end_entity;
```

Отметим, что между символами (* и *) записывается *комментарий* — произвольный текст по усмотрению автора модели. Если комментарий умещается в одной строчке, то достаточно перед его текстом поставить двойной дефис (--).

Простые типы данных. К *простым типам данных* относятся следующие типы:

— **integer** — целые числа;
— **real** — вещественные числа;
— **number** — тип, объединяющий типы **integer** и **real**;
— **logical** — его значениями могут быть **true, false** или **unknown** (неопределенность);
— **Boolean** — с возможными значениями **true** или **false**;
— **binary** — последовательность битов 1 или 0;
— **string** — строка символов.

Их изображения на схемах на языке Express-G показаны на рис. 6.23.

Для **binary** и **string** в круглых скобках можно указать максимально возможное число элементов множества, например, если строка A может включать до 24-х символов, то:

A: **string**(24);

если ровно 24 символа, то

A: **string**(24) **fixed;**

если ограничений нет, то

A: **string;**

BINARY NUMBER REAL
LOGICAL BOOLEAN GENERIC
INTEGER STRING

Рис. 6.23. Изображение простых типов данных в Express-G

Если переменная a имеет тип **binary**, то выражение a[5:7] означает биты с 5-го по 7-й в коде a.

Значения простых типов выражаются с помощью литералов. *Литералы* — это числа (целые, вещественные), двоичные коды, логические значения (**true, false**, **unknown),** фрагменты текста (строковый тип). Примеры записи литералов:

двоичный (начинается с знака %)	%100101110
целое десятичное число	1052
вещественный (обязательна десятичная точка)	34.e-3 или 0.034
строковый (занимает не более одной строки)	‘first name’

Агрегативный тип данных. *Агрегативный тип данных* — множество элементов некоторого типа.

Различают четыре разновидности агрегативных типов, сведения о которых приведены в табл. 6.1.

Таблица 6.1

Тип данных	Упорядоченность	Различие элементов
array	Да	Необязательно
bag	Нет	Необязательно
list	Да	Обязательно
set	Нет	Обязательно

При описании типа *array* после слова **array** в квадратных скобках указываются нижняя и верхняя границы индексов. Для остальных агрегативных типов записываются не граничные значения индекса, а нижняя и верхняя границы числа элементов. Например:

```
F1: array[2:8] of real;
(*описание семиэлементного массива F1, его элементы имеют тип real и нумеруются, начиная с значения индекса 2*)
F2: list[1:?] of integer;
(*множество F2 содержит, по крайней мере, один элемент типа integer*)
matr: array[1:10] of array[9:12] of atrac;
(*массив matr состоит из 10 четырехэлементных массивов, элементы типа atrac*)
```

Записи вида **array**[2:8] или **list**[1:?] в Express-G преобразуются в форму **A**[2:8] или **L**[1:?], указываемую около линии атрибута агрегативного типа после имени этого атрибута. Так, первый из вышеприведенных примеров представлен на рис. 6.24.

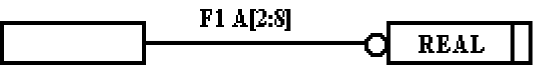


Рис. 6.24. Обозначение атрибута агрегативного типа в Express-G

Определяемый, нечисловой, выделяемый типы. *Определяемый тип* обычно вводится пользователем для улучшения читаемости модели. *Нечисловой тип* — тип данных, экземплярами которого являются нечисловые (предметные) переменные. *Выделяемый тип* соответствует поименованной совокупности других типов. Описание этих типов данных начинается со служебного слова **type,** за которым следует идентификатор типа и его определение. Пример описания определяемого типа:

```
type volume = real;
end_type;
entity manual;
name: string;
v1,v2,v3: volume;
end_entity;
```

Определение нечислового типа начинается со служебных слов **enumeration of,** после которых в скобках перечисляются элементы множества. Например:

```
type color = enumeration of (red, green, blue);
end_type;
```

Ссылка на значение red теперь возможна в виде red или color.red.

Выделяемый тип соответствует одному из некоторого списка уже введенных типов. Этот список записывается после служебного слова **select.** Ссылка на имя выделяемого типа означает, что выбирается один из типов совокупности:

```
type a_c = select (one, two, three);
end_type;
. . .
proc: a_c; (* proc может быть объектом одного из типов one, two, three*)
```

Графические изображения определяемых, нечисловых и выделяемых типов данных показаны на рис. 6.25. Внутри прямоугольников, ограничиваемых пунктирными линиями, записывается имя типа.

Супертипы и подтипы. Отношения типа целое-часть или функция-вариант реализации, характерные для представления структур объектов в виде альтернативных (И-ИЛИ) деревьев, в языке EXPRESS выражаются в форме отношений между типами данных. Для этого введены понятия *супертипа (supertype)*, как более общего типа, и *подтипов (subtypes)*, как подчиненных типов. На рис. 6.26 верхняя сущность относится к супертипу, а три нижних прямоугольника изображают подтипы, линии связи прямоугольников должны быть утолщенными.

Рассмотрим пример фрагмента И-ИЛИ-дерева, в котором имеется ИЛИ вершина a1 и две подчиненные ей альтернативные вершины b1 и b2. Общим атрибутом для b1 и b2 является size типа **real**, специфичный для b1 атрибут — vol типа **real**, а специфичный для b2 атрибут met типа **string**. Этот фрагмент может быть описан следующим образом:

```
entity a1
supertype of (oneof (b1,b2));
size: real;
end_entity;
entity b1
subtype of (a1);
vol: real;
end_entity;
entity b2
subtype of (a1);
met: string;
end_entity;
```

defined
enumeration
select

Рис. 6.25. Изображения типов данных в Express-G

Рис. 6.26. Изображение иерархии типов в Express-G

Используются также следующие правила записи супертипов и подтипов:

— в случае, если a1 есть И вершина, вместо **oneof** используется зарезервированное слово **and** (в более общем случае **andor**), т.е. вторая строчка примера будет выглядеть так:

```
supertype of (b1 and b2);
```

— если между подтипами нет взаимосвязи, выражаемой логической функцией (в частности, ИЛИ или И вершинами), то указание в a1 факта, что это супертип, не требуется; достаточно упоминание о подчиненности подтипов в их декларациях в виде **subtype of** (a1);

— перед декларацией **supertype** записывается зарезервированное слово **abstract**, если вершине a1 не соответствуют какие-либо экземпляры сущности, т.е. если a1 введена только для указания общих для подтипов атрибутов;

— у одного подтипа может быть больше одного супертипа; подтип наследует атрибуты всех своих супертипов; если в декларациях супертипов используются одинаковые идентификаторы атрибутов, то ссылка на них должна быть в виде составного идентификатора, например, a1.size.

Пример:

```
entity device
supertype of (oneof (transistor, diode));
(* device есть ИЛИ вершина И-ИЛИ-дерева с двумя альтернативами transistor и diode*)
end_entity;
entity transistor —
subtype of (device);
b: real;
end_entity;
entity diode
subtype of (device);
r: real;
end_entity;
```

Ограничения. Ограничения, накладываемые на экземпляры сущности, выражаются с помощью *правил (rules)*. Правила могут быть общими или локальными.

Описание правила, общего для ряда сущностей, начинается со служебного слова **rule**, далее следуют идентификатор правила, служебное слово **for**, ссылки на сущности, на которые правило распространяется, и, наконец, собственно ограничения.

Локальные правила могут описывать ключевые атрибуты (*uniqueness rules*) или выражать ограничения, накладываемые на атрибуты некоторой сущности (*domain rules*). Например, если ключевой атрибут сущности Z есть составной атрибут X.Y, или, другими словами, одному сочетанию значений атрибутов X и Y должен соответствовать единственный экземпляр сущности Z, то

```
entity Z;
X: integer;
Y: string;
unique
X,Y;
end_entity;
```

Ограничение на атрибуты некоторой сущности выражается с помощью правила в теле этой сущности. Ограничение записывается после слова **where** в виде выражения, значениями которого могут быть **true, false** или **unknown**. Допустимыми значениями атрибута будут только те, для которых выражение принимает значение **true**. Например, можно записать, что длина вектора vect = (x,y,z) должна быть равна единице, в виде правила cons:

```
entity vect;
x,y,z: real;
where
cons: x**2 + y**2 + z**2 = 1.0;
end_entity;
```

Процедуры и функции. *Процедуры* и *функции* служат для описания процедурной части модели. Как и в алгоритмических языках, используется концепция формальных и фактических параметров. Описание процедуры начинается с служебного слова **procedure**, за которым следуют идентификатор процедуры и описание формальных параметров в круглых скобках. Пример описания заголовка процедуры:

```
procedure eq (x,y: real; n: integer; var result: route);
```

Аналогично описываются функции, их отличает только описание в заголовке типа результата после закрывающей скобки:

```
function log (a: real; m: integer): real;
```

Локальные переменные, описанные в блоке

```
local
...
end_local;
```

действуют только в пределах данных функции или процедуры.

Ряд функций и процедур относится к стандартным и потому не требует описания во вновь разрабатываемых моделях. Отметим следующие стандартные функции:

Abs — абсолютная величина; **Sqrt** — корень квадратный; **Exp** — экспонента; **Log, Log2, Log10** — логарифмы натуральный, двоичный, десятичный соответственно; **Sin, Cos, Tan, Acos, Asin, ATan** — тригонометрические и обратные тригонометрические функции *sin, cos, tg, Arc cos, Arc sin, Arc tg*.

В число стандартных входят также функции: **BLength** — подсчет числа бит в двоичном коде; **HiBound** — верхняя граница индекса у *array* или верхняя граница числа элементов у *set, bag, list*; **LoBound** — то же в отношении нижних границ; **Length** — подсчет числа символов в строке; **Odd** — возвращает значение true, если аргумент — нечетное число; **SizeOf** — возвращает число элементов в объекте агрегативного типа; **TypeOf —** возвращает список типов, к которым принадлежит параметр этой функции; **Exists** — возвращает значение true, если аргумент этой функции входит в число атрибутов соответствующей сущности, и др.

К стандартным процедурам относятся процедуры **Insert** и **Remove** — вставка или изъятие элемента в заданной позиции у объекта агрегативного типа соответственно.

При описании алгоритмов в телах процедур и функций могут использоваться операторы *пустой (Null)*, *присваивания (Assignment)*, *выбора (Case)*, *составной (Compound Statement)*, *условный (if..then..else)*, *цикла (Repeat)*, *выхода из функции или процедуры (Return)*, *перехода на конец цикла (Skip)*.

В выражениях используются операнды, знаки операций, вызовы функций. Так, для арифметических операций над числами типа **real** применяются следующие знаки: * — умножение, / — деление, DIV — целочисленное деление, + — сложение, — - вычитание, ** — возведение в степень, MOD – деление по модулю.

Знаки логических операций: **not** — отрицание, **and** — конъюнкция, **or** — дизъюнкция, **xor** — исключающее ИЛИ. В применении к величинам типа **logical** эти операции выполняются по правилам действий в трехзначном алфавите. Логическое выражение a1 **in** a2 принимает значение **true**, если a1 содержится в a2. Оператор **like** используется для посимвольного сравнения строк. Для сравнения экземпляров сущностей используют операции “равно” и “неравно” со знаками :=: и :<>: соответственно.

Операции над множествами (типами **bag** и **set**) – пересечение (*Intersection*), объединение (*Union*), разность (*Difference*). Их знаки суть * (умножение), + (плюс), — (минус) соответственно. Оператор **Query** (A <* B | C) возвращает подмножество тех элементов из агрегативного типа B, для которых выполняется условие C, здесь A – простая переменная, используемая в C. Знак + (плюс) по отношению к операндам типа **binary** или **string** есть знак конкатенации.

В качестве формальных параметров процедур и функций, кроме типов данных, применяемых в других конструкциях языка и охарактеризованных выше, могут использоваться обобщенные типы: *generic, aggregate* и некоторые другие. Тип *generic* формального параметра означает, что соответствующий фактический параметр может иметь любой тип данных из числа предусмотренных при описании процедуры. Аналогично тип *aggregate* обобщает агрегативные типы данных — *array, bag, list, set*. Например:

```
function add (a,b: generic: intype): generic: intype;
local
nr: number;
vr: vector;
end_local;
if (‘number’ in typeof (a)) and (‘number’ in typeof (b))
then nr := a+b;
(* функция typeof (a) возвращает тип аргумента a и, если этот тип есть number, то первый операнд
  логического выражения равен true *)
return (nr);
else
if (‘this schema.vector’ in typeof (a)) and (‘this schema.vector’ in typeof (b)) then
vr.i := a.i + b.i;
vr.j := a.j + b.j;
vr.k := a.k + b.k;
(* подразумевается, что декларация типа vector была произведена в схеме с именем this schema *)
return (vr);
end_if;
end_if;
end_function;
```

В языке Express-G специальные символы для изображения правил, процедур и функций не оговорены.

Константы. Способ описания констант очевиден из следующего примера:

```
constant
year: integer:= 1995;
start: date := date(12,16,1982);
(*подразумевается, что при описании типа date указаны три атрибута месяц, число, год*)
end_constant;
```

Взаимосвязи схем. Для установления интерфейса между двумя схемами вводятся спецификации интерфейса. Применяют два типа спецификаций — **use** и **reference.** Например:

```
schema s1;
entity par1;
name: string;
end_entity;
end_schema;
schema s2; (* в схеме s2 в качестве параметра x используется name из s1.par1 *)
use from s1.par1 ( name as x);
end_schema;
```

Ссылки типа use отличаются тем, что декларации сущностей из другой схемы используются в данной схеме как свои локальные, в то время как reference просто позволяет обращаться к декларациям другой сущности.

В языке Express-G используются диаграммы двух уровней. На схемном уровне (schema level) изображаются схемы и их взаимосвязи в виде линий. На сущностном уровне (entity level) изображаются типы, сущности, атрибуты, а для ссылок на объекты другой схемы применяются специальные символы.

Эти символы представляют овальными фигурами. В овале записывают имя схемы-источника и имя используемого определения. В нашем примере это ссылка на S1.par1. Овал помещается внутрь прямоугольника, в котором дополнительно указывается имя атрибута (в примере это name).

Для указания межстраничной связи, что требуется, если Express-G модель размещается более чем на одной странице, используется овальный символ, внутри которого указываются через запятую номер страницы и номер ссылки.

Пример модели на языке Express. Пример *"person_organization_schema"* взят из 41-го тома "Интегрированные ресурсы" стандарта STEP (ISO 10303.41).

```
schema person_organization_schema;
entity address;
internal_location : optional label;
street_number : optional label;
street : optional label;
postal_box : optional label;
town : optional label;
region : optional label;
postal_code : optional label;
country : optional label;
facsimile_number : optional label;
telephone_number : optional label;
electronic_mail_address : optional label;
telex_number : optional label;
where
wr1 : exists(internal_location) or exists(street_number) or exists(street) or exists(postal_box) or
  exists(town) or exists(region) or exists(postal_code) or exists(country) or exists(facsimile_number)
  or exists(telephone_number) or exists(electronic_mail_address) or exists(telex_number);
end_entity;
entity personal_address
subtype of (address);
people : set[1:?] of person;
description : text;
end_entity;
entity person;
id : identifier;
```

```
last_name : optional label;
first_name : optional label;
middle_names : optional list[1:?] of label;
prefix_titles : optional list[[1:?] of label;
suffix_titles : optional list[[1:?] of label;
unique
ur1 : id;
where
wr1 : exists(last_name) or exists(first_name);
end_entity;
end_schema;
```

Структура обменного файла в стандарте STEP (ISO10303-21). Внутри стандарта STEP введен обменный файл, но технология обмена данными между различными системами более полно разработана в стандарте P-LIB. Обменный файл в STEP состоит из головной и информационной секций. В головной секции (между служебными словами HEADER и ENDSEC) указываются:

Entity file_name — имя и некоторые другие атрибуты данного конкретного обменного файла;

Entity file_description — неформальное описание содержимого файла и требования к ПО для обработки данного файла;

Entity file_schema — схемы, для которых далее даны экземпляры сущностей;

keyword (список типов).

В информационной секции (между словами DATA и ENDSEC) указываются имена экземпляров сущностей и значения их атрибутов в виде следующих строк:

имя сущности = keyword (список параметров);

Например:

```
#1 = POINT(0.0,0.2,0.5);
(* экземпляр сущности типа POINT с именем 1 имеет значения параметров 0, 0.2 и 0.5 типа REAL.*)
#2 = WIDGET(.RED.);
(* экземпляр сущности типа WIDGET с именем 2 имеет значение перечислимого типа RED.*)
. . .
#8 = LINE(#1,#4);
(* значениями атрибутов являются экземпляры сущностей с именами 1 и 4.*)
```

В списке параметров значения перечисляются в том же порядке, в каком они фигурировали в описании сущности.

Расширения языка Express. В языке Express-C добавляются возможности описания событий и транзакций:

```
event a;
when b ⇒ c; (* здесь b — логическое выражение, c — обращение к транзакции при b  =true*);
end_event;
transaction c;
local d: e;
end_local;
. . .
end_transaction;
```

При описании соответствия между двумя Express-моделями используются языки Express-X или Express-M. Например, в Express-M соответствие между схемой-источником A, в которой заданы атрибуты a1, a2, a3, и схемой-целью B, в которой те же атрибуты описаны идентификаторами b1, b2, b3, выражается следующим описанием:

```
shema map B ← A;
b1 := a1;
b2 := a2;
b3 := a3;
end_shema_map;
```

При отображении возможны преобразования атрибутов, например, если a1 задан в метрах, а b1 в сантиметрах, то в примере нужно записать b1 := a1*100.

Упражнения и вопросы для самоконтроля

1. Назовите основные стадии проектирования технических систем. Для чего нужно прототипирование?
2. Что такое “профиль открытой системы”?
3. Чем обеспечивается открытость систем?
4. Что понимают под диаграммой потока данных?
5. Представьте IDEF0-диаграмму верхнего уровня для этапов жизненного цикла промышленной продукции.
6. Приведите пример неспецифического отношения.
7. Постройте IDEF1X-диаграмму для сущностей “студенческая группа, студент, преподаватель, дисциплина”.
8. Назовите причины появления стандартов STEP.
9. Что называют прикладным протоколом в STEP-технологии?
10. Что понимается под метамоделью в CASE системах?
11. Представьте диаграмму, полученную в п.7 на языке Express.
12. Опишите назначение и структуру обменного файла в языке Express.
13. Для чего нужны разновидности языка Express, такие как Express-M и Express-V?



Методики IDEF

Таблица П.1

Название	Назначение
IDEF0	Функциональное моделирование (Function Modeling Method)
IDEF1, IDEF1X	Информационное моделирование (Information and Data Modeling Methods)
IDEF2	Поведенческое моделирование (Simulation Modeling Method)
IDEF3	Моделирование процессов (Process Flow and Object State Description Capture Method)
IDEF4	Объектно-ориентированное проектирование (Object-Oriented Design Method).
IDEF5	Систематизация объектов приложения (Ontology Description Capture method)
IDEF6	Обоснование проектных действий (Design Rationale Capture Method)
IDEF8	Взаимодействие человека и системы (Human-System Interaction Design)
IDEF9	Выявление и учет условий и ограничений проектирования (Business Constraint Discovery)
IDEF14	Моделирование вычислительных сетей для АС (Network Design)

Стандарты CDIF (CASE Data Interchange Format)

CDIF-стандарты подразделяют на три группы.

А. Структура стандартов.

“CDIF CASE Data Interchange Format — Overview” EIA/IS-106; общая концепция стандартов CDIF.

“CDIF / Framework for Modeling and Extensibility” EIA/IS-107; поскольку стандарты CDIF должны быть применимы для экспорта/импорта данных в различных предметных областях, а существующие стандарты могут не отражать особенностей некоторых предметных областей, то требуются средства для включения в CDIF новых частей, описывающих эти особенности. Данный стандарт определяет инструментальные средства для включения описаний новых предметных областей.

Б. Формат обмена данными.

“CDIF Transfer Format / General Rules for Syntaxes and Encodings” EIA/IS-108; стандарт устанавливает правила для включения в CDIF форматов с другим синтаксисом или других способов кодирования.

“CDIF Transfer Format / Transfer Format Syntax SYNTAX.1” EIA/IS-109; основной синтаксис, принятый в настоящее время.

“CDIF Transfer Format / Transfer Format Encoding ENCODING.1” EIA/IS-110; основной способ кодирования, принятый в настоящее время.

“CDIF Transfer Format / OMG IDL Bindings”; предназначен для передачи данных по Internet с использованием средств CORBA.

В. Интегрированная метамодель.

“CDIF — Integrated Meta-model / Foundation Subject Area” EIA/IS-111; описывает иерархическое представление информации с использованием наследования.

“CDIF — Integrated Meta-model / Common Subject Area” EIA/IS-112; описывает свойства, инвариантные для многих объектов. Определенные здесь свойства далее наследуются в описании семантики и формы представления. Примеры таких свойств: ограничения, имена и т.п.

“CDIF — Integrated Meta-model / Data Modeling Subject Area” EIA/IS-114; задает основные формы моделей “отношения-атрибуты”.

“CDIF — Integrated Meta-model / Data Flow Model Subject Area” EIA/IS-115; поддерживает моделирование потоков данных с использованием декомпозиции процессов, источников, потоков по методикам таким, как IDEF0.

“CDIF — Integrated Meta-model / Data Definition Subject Area”; задает описание наиболее общих атрибутов таких, как простые типы данных, деньги, даты и т.п.

“CDIF — Integrated Meta-model / State Event Model Subject Area”; поддерживает методики, оперирующие состояниями, переходами между ними, событиями, условиями и действиями, т.е. методики поведенческого моделирования.

“CDIF — Integrated Meta-model / Object-Oriented Analysis and Design Subject Area”; поддерживает современные объектно-ориентированные методики и средства такие, как методики Буча или UML. Этот стандарт рассматривается, как относящийся к наиболее важным.

“CDIF — Integrated Meta-model / Computer Aided Control Systems Design Subject Area”; концепция управ-

ления проектированием в реальном масштабе времени.

“CDIF — Integrated Meta-model / Presentation Location and Connectivity Subject Area” EIA/IS-118; поддерживает представление моделей в форме графа (узлы, ребра, присоединенные к ним метки).

“CDIF — Integrated Meta-model / Physical Relational Database Subject Area” — стандарт относится к физическому уровню реляционных баз данных.

“CDIF — Integrated Meta-model / Project Management Planning and Scheduling Subject Area” — стандарт относится к предметной области, связанной с управлением планированием и потоками работ.

“CDIF — Integrated Meta-model / Business Process Modeling Subject Area” — стандарт относится к предметной области моделирования бизнес процессов.

Стандарты STEP

Третья группа стандартов (тома с 31 по 39) в ISO 10303 относится к методам тестирования продукции.

***N*=31:** *General concepts*; в стандарте представлены общие положения определения соответствия продукции требованиям стандартов STEP, включая организацию тестирования (аккредитацию, сертификацию, взаимоотношения между контролерами и клиентами).

***N*=32**: *Requirements on testing laboratories and clients*; устанавливаются требования к контролерам и клиентам.

***N*=33**: *Structure and use of abstract test suites*; устанавливаются структура и порядок использования тестовых последовательностей.

***N*=34**: *Abstract test methods*; описываются методы тестирования.

***N*=35**: *Abstract test methods for SDAI implementations*; то же по отношению к SDAI представлениям.

Группа стандартов “Набор средств тестирования” включает тома, начинающиеся с номера 301. Каждый из этих стандартов соответствует прикладному протоколу, имеющему уменьшенный на 100 номер, т.е. стандарт с *N*=301 определяет средства тестирования для объектов протокола AP201 и т.д.

Группа стандартов «Интегрированные общие ресурсы» (*N* от 41 до 49) описывает модели, используемые во многих прикладных протоколах.

***N*=41**: *Fundamentals of product description and support*; основы описания и поддержки изделий. В протоколе определяются такие понятия и группы сущностей, как продукт, представление формы (shape_representation), операция (action), контекст — аспект описания (application and product context), статус утверждения (approval), контракт, дата, типы документов, исполнители (организации и персоналии), единицы измерения длин, площадей, масс, температур и др.

***N*=42**: *Geometric and topological representation*; представление геометрии и топологии. В стандарте определен ряд сущностей, их набор близок к набору, используемому в таком стандарте, как IGES. В частности, используются следующие понятия: положение координатной оси (axis_placement), модели кривых в форме B-сплайна (b_spline_curve) и Безье (bezier_curve), модели поверхности в форме B-сплайна (b_spline_surface), рационального B-сплайна (rational_b_spline_surface) и Безье (bezier_surface), точка в декартовых координатах (cartesian_point), преобразование декартовых координат (cartesian_transformation_operator_3d), геометрический аспект (geometric_representation_context), полигональная поверхность (offset_surface), поверхность вращения (surface_of_revolution) и др.

***N*=43**: *Representation structures*; структуры представления. Средства описания аспектов документации, ее атрибутов, составных частей и т.п.

***N*=44**: *Product structure configuration*; структурирование изделий.

***N*=45**: *Materials*; представлены свойства материалов.

***N*=46**: *Visual Presentation*; визуализация. Стандарт построен на базе положений, ранее принятых в стандартах GKS (Graphic Kernel System) и PHIGS (Programmer’s Hierarchical Interactive Graphic System). Вводятся группы терминов, относящихся к представлению (presentation), визуализации (visualization), цвету и др.

***N*=47**: *Shape variation tolerance*; допуски.

***N*=48**: *Form features*; свойства форм.

***N*=49**: *Process structure and properties*; структура процессов и свойства.

Более специфические прикладные ресурсы выделены в протоколы с номерами, начиная с *N* = 101.

***N*=101**: *Draughting*; определяются сущности, относящиеся к процедурам черчения.

***N*=104**: *Finite element analysis*; анализ по методу конечных элементов. Описание стандарта на языке Express состоит из нескольких схем. В одной из них задаются геометрические аспекты модели. Здесь описываются следующие типы данных: система координат (декартова, цилиндрическая, сферическая); виды конечных элементов (объемный, поверхностный 2*D* или 3*D,* участок 2*D* или 3*D* кривой), форма элемента (линейный, квадратичный, кубический); степень свободы; параметры и дескрипторы элементов, позиция элемента, свойства

элементов (например, масса, момент инерции, жесткость), материал и его свойства (плотность, эластичность, тепловое расширение), группа элементов и др. В другой схеме основное внимание уделено математическим представлениям. Например, здесь фигурируют такие сущности и типы данных, как тензоры; переменные, характеризующие напряжения; приложенные нагрузки; погрешности; шаги анализа и т.п.

***N*=105**: *Kinematics*; кинематика.

***N*=106**: *Building construction core model*; основная модель строительных конструкций.

***N*=107:** *Engineering analysis core ARM*; ядро инженерного анализа.

Прикладные протоколы. Как правило, прикладной протокол включает большое число сущностей и их атрибутов, описания AP составляют десятки страниц на языке Express или десятки рисунков на языке Express-G. Поэтому целесообразно использовать приемы группирования тесно взаимосвязанных сущностей для более лаконичной характеристики AP. Такими группами являются единицы функциональности (UoF — Units of Functionality). Используют также понятие классов (CC – Conformance Classes) для классификации используемых моделей.

AP201: *Explicit draughting*; явное черчение. При использовании протокола оперируют такими понятиями, как структура чертежа, аннотация, геометрическая форма детали, группирование. В число сущностей входят спецификация, утверждение, номер листа, организация-исполнитель, слой, вид и т.п.

AP202: *Associative draughting*; ассоциативное черчение. Протокол, относящийся к описанию конструкторской документации. В протоколе фигурируют данные, в значительной мере пересекающиеся с данными протокола AP201 и сгруппированные по UoF следующим образом:

1) структура документации (иерархия, заголовки, утверждающие подписи);
2) связь с изделием (версия, изготовитель);
3) аннотация формы изделия (2D или 3D CAD-модель);
4) связь модели с ее визуализацией (масштаб);
5) форма аннотации (месторасположение аннотации, символы, заполняемые позиции);
6) оформление документов (шрифты, цвета);
7) размеры (допуски);
8) группирование деталей по тем или иным признакам.

AP203: *Configuration controlled design*; проектирование с конфигурационным управлением. Это один из важнейших прикладных протоколов. В нем унифицированы геометрические модели, атрибуты и спецификации: сборок; 3D поверхностей, разделенных на несколько классов; параметры управления версиями и внесением изменений в документацию и др.

Описание протокола AP203 на языке Express представляет собой схему, в которой можно выделить следующие части.

1. Ссылки на заимствованные из стандартов ISO 10303-41, 42 и 44 интегрированные ресурсы. Это ссылки на такие сущности, как контексты приложения и продукции, свойства изделий, массо-габаритные характеристики, расположение координатных осей, типы кривых и поверхностей, указатели статуса контракта, предприятия, исполнителей, даты и т.п.

2. Описания некоторых обобщенных типов, объединяющих с помощью оператора SELECT ряд частных типов.

3. Описания сущностей, выражающих конструкции изделий. Представлены 6 классов геометрических моделей. *Класс 1* предназначен для задания состава изделий без описания геометрических форм. *Класс 2* включает каркасные модели с явным описанием границ, например, в виде координат точек и определяемых с их помощью линий. В *классе 3* каркасные модели дополнены топологической информацией, т.е. данными о том, как поверхности, линии или точки связаны друг с другом. *Класс 4* служит для описания поверхностей произвольной формы. *Классы 5 и 6* включают твердотельные модели, так называемые BREP (Boundary representation). К первому из них относятся тела, границы которых аппроксимированы полигональными (фасеточными) поверхностями, состоящими из плоских участков. В классе 6 поверхности, ограничивающие тела, могут быть как элементарными (плоскими, квадратичными, тороидальными), так и представленными моделями в форме Безье, В-сплайнов и др.

4. Описание других используемых сущностей, относящихся к конфигурации изделия, например, таких как вносимое в проект изменение с соответствующими атрибутами.

AP204. *Mechanical design using boundary representation*; конструирование механических деталей на основе твердотельной модели. В протоколе введены такие сущности, как имя изделия, шифр, версия, сборочный узел, модель (элементарная, фасеточная или универсальная BREP-модель), цвет, ширина линий представления и т.п.

AP205: *Mechanical design using surface representation*; конструирование механических деталей на основе

поверхностной модели. Ряд понятий, используемых в этом протоколе, аналогичен понятиям протокола AP204, но используются поверхностные модели с границами, представленными геометрически или топологически.

AP207: *Sheet metal die planning and design*; проектирование листовой штамповки.

AP208: *Life cycle management — Change process*; управление процессами изменений в жизненном цикле (управление конфигурацией). Включает идентификацию фактов (недостатков), требующих внесения изменений, их причин, определяет действия по устранению недостатков и лиц, вносящих изменения.

AP209: *Composite and metallic structural analysis and related design*; анализ композитных и металлических конструкций; комбинирование данных геометрии и управления конфигурацией с данными для анализа, например, по методу конечных элементов. Поддерживаются статический и частотный анализ, 3*D* сеточные модели для анализа по МКЭ, вводятся определения свойств сборок, средства для представления свойств композитных и однородных материалов.

AP210: *Electronic assembly, interconnect and packaging design*; компоновка и проектирование межсоединений в электронной аппаратуре, управление конфигурацией и представление данных о печатных платах и сборках при их проектировании и при передаче данных на производственную стадию. В протоколе используются данные о форме и материале изделия, размещении компонентов и имеющихся ограничениях, проводящих и изолирующих слоях, вносимых изменениях в проект и т.д.

AP211: *PCA Integrated Diagnositics and test*; тестирование и диагностика электронной аппаратуры.

AP212: *Electrotechnical design and installation*; проектирование и монтаж электротехнических изделий. В протоколе описываются электротехнические системы на стадиях проектирования, монтажа, поставки. Имеются средства для представления функциональной декомпозиции систем, физического размещения оборудования и кабельных соединений, информационного обмена между частями систем, документирования, управления конфигурацией и др. (Но в протоколе не рассматриваются вопросы изготовления, моделирования, тестирования аппаратуры). Примеры используемых в стандарте объектов: электротехнические системы и приборы, функциональный продукт, место размещения (installation_location), сигнал, терминал, проект, контракт, интерфейс, цепь, соединение (connectivity), порт. Отдельную группу составляют объекты, представляемые графически, и др.

В протоколе описывается ряд опций, которые могут быть использованы в моделях. Состав этих опций зависит от класса формы. Всего в протоколе 4 класса (CC — conformance classes):

CC1 — проектные данные (классификация, конфигурация, документация с двумерными схемами, структура) без функциональных аспектов и инсталляции;

CC2 – класс 1 с добавлением функциональной информации (распределение функций между частями системы, информационные потоки и др.);

CC3 — класс 1 с информацией об инсталляции (двумерные чертежи с геометрической и пространственной информацией, схемы размещения оборудования);

CC4 – полная совокупность данных – единиц функциональности протокола AP212, т.е. объединение CC1, CC2 и CC3.

AP213: *Numerical control process plans for machined parts*; проектирование обработки на оборудовании с числовым программным управлением. В протоколе введены средства для описания производственных операций, технологического оборудования и инструментов, материалов, геометрических форм и допусков изделий, рабочих мест, сопроводительных административных данных.

AP214: *Core Data for Automotive Mechanical Design Processes*; основные данные для проектирования механических частей автомобилей. Имеются средства для представления данных по структуре и геометрии изделий, презентации проектов, моделированию, производственным процессам (числовое управление, допуски, материалы) и др.

В стандарте введено 19 классов моделей (Conformance Classes), классы различаются видом модели (поверхностная, твердотельная, каркасная), наличием данных по кинематике, допускам, управлению конфигурацией.

Геометрические группы родственных понятий (сущностей, атрибутов), фигурирующих в приложении, сведены в AP214 в несколько UoF, имеющих непустые пересечения. Это:

G1: wireframe_model_2*d*, включающая такие сущности, как геометрическая модель, точка, линия, кривая, гипербола, B-сплайн, 2*D* каркасная модель и др.;

G2: wireframe_model_3*d* с аналогичными сущностями, но в пространстве 3*D*;

G3: connected_surface_model, предназначена для представления топологически ограниченных поверхностных моделей, эта группа включает ряд сущностей из G2 и G8 и таких, как кривая или точка на поверхности, цилиндрическая и тороидальная поверхности, конструктивная геометрия и др.

G4: faceted_b_rep_model, относится к BREP моделям с деталями, имеющими планарные поверхности и

внутренние пустоты. Понятия точки, линии, плоскости взято из G3 и G5, другие сущности – замкнутая фасеточная оболочка, твердотельное BREP многообразие (manifold solid B-rep) и др.

G5: b_rep_model — представление одного или более тел, каждое из которых состоит из замкнутых внешней и внутренних оболочек. Геометрия поверхностей выражена кривыми. Большинство понятий аналогично используемым в G3.

G6: compound_model — модели поверхностные, твердотельные, каркасные с топологически представленными соединениями. Примеры использования: выделение в телах зон с различными свойствами, частей сварной конструкции и т.п.

G7: csg_model, или более полное название solid model using Constructive Solid Geometry – получение модели с помощью булевых операций над заданными телами. Наряду с понятиями из предыдущих UoF здесь фигурируют понятия блок, примитив, результат булевой операции и др.

G8: geometrically_bounded_surface_model UoF – геометрически ограниченная поверхностная модель.

Среди других UoF можно отметить:

S2: element_structure — элементы структуры и аннотаций структуры, например, слой, образец, аспект формы, преобразование 2D или 3D, точность, расположение осей и т.п.

S5: work_management с такими сущностями, как операция, метод операции, контракт, порядок работ, изменение.

S6: classification с понятиями классификации атрибутов и систем, иерархии и пунктов классификаций.

S7: specification_control — управление спецификациями предназначено для описания свойств продуктов, имеющих большое число вариантов. Описываются классы продуктов, категории характеристик, способы декомпозиции продукции, ее функции, вводятся сущности конфигурация, проектное ограничение, проектное решение, пункт решения, вариант размещения, спецификация и т.п.

AP215. *Ship arrangement*: расположение частей судна. Затрагиваются такие аспекты, как декомпозиция на пространственно выделенные части (например, грузовые отсеки, машинное отделение, каюты, переборки), форма корпуса, водоизмещение и т.п.

AP216. *Ship moulded form*; форма судна. Описываются общие характеристики, размеры, гидростатика, протяженные внутренние поверхности, геометрия надстроек.

AP217. *Ship piping;* система судовых трубопроводов. Протокол относится к геометрии трубопроводов, их прочности, материалам, анализу потоков, управлению конфигурацией при их проектировании.

AP218. *Ship structures*; устройство судна. В этом приложении рассматриваются характеристики внутреннего устройства судна.

AP220: *PCA Manufacturing Planning*; производство печатных плат и сборок. Вводятся средства для представления 2*D* геометрии размещения компонентов, допусков, операций изготовления, измерения_и т.п.

AP221: *Functional data and their schematic representation for process plant*; функциональная модель и ее схемное представление для производственных процессов. Протокол предназначен для описания иерархического построения предприятий химического, нефтеперерабатывающего производства, атомной энергетики. Рассматриваются состав оборудования, система трубопроводов, характеристики потоков в них.

AP223: *Exchange of design and manufacturing product information for casting parts*; обмен проектными и технологическими данными для литейного производства. В протоколе предусмотрены следующие аспекты приложения: литье в песчаные формы, моделирование процессов литья, литейное оборудование и материалы, процессы плавления, заливки, охлаждения, экстракции, контроль и тестирование.

AP224: *Mechanical product definition for process plans using machining features*; описание механических деталей для планирования обработки. Имеются средства для описания особенностей конструкции деталей (например, отверстий, бобышек, буртов), требований к качеству . обработки, свойств материалов, геометрической формы и др.

AP225: *Building elements using explicit shape representation*; элементы строительных конструкций с явным представлением их формы.

AP226. *Ship mechanical systems*; корабельное механическое оборудование. С помощью определений этого протокола описываются силовые установки, электрогенераторы, насосы, компрессоры, соединения компонентов, их функции, параметры шума и вибраций и т.п.

AP227. *Plant spatial configuration*; пространственная конфигурация предприятий.

AP228: *Building services: Heating, ventilation, and air conditioning*; инженерные службы в строительстве — теплоснабжение, вентиляция, кондиционирование воздуха.

AP231. *Process design and process specifications of major equipment*; проектирование и описание основного оборудования. Протокол относится к концептуальному проектированию, контролю, моделированию, мате-

риалам оборудования предприятий химической и нефтегазовой промышленности.

AP232. *Technical data packaging core information and exchange*; представление и обмен технических данных. Протокол посвящен взаимодействию систем управления данными разных проектирующих систем. Объектами описания служат проектные данные как выраженные средствами прикладных протоколов, так и не соответствующие стандартам STEP. Это чертежи, программы для оборудования с ЧПУ, модели проектируемых объектов, спецификации, бизнес-документация и др.

AP233. *Systems engineering data representation* – системы представления инженерных данных Имеются в виду данные (единицы функциональности), характеризующие состояния системы и ее параметры (например, цена, производительность, надежность, технологичность, контролепригодность и т.п.), связанные с требованиями к продукту, его функциональной архитектурой, поведением, управлением конфигурацией. Рассматриваются как количественные, так и лингвистические (в том числе нечеткие) переменные вместе с единицами измерения.

Ряд протоколов, начинающийся с $N = 501$, содержит геометрические модели и часто используемые элементы чертежей. Среди них:

***N*=501**: *Edge-based wireframe*; каркасная модель на основе граней..

***N*=502**: *Shell-based wireframe*; каркасная модель на основе оболочек.

***N*=503** (CD): *Geometrically bounded 2D wireframe*; 2D каркасная модель с геометрически заданными границами.

***N*=504**: *Draughting annotation*; чертежные аннотации.

***N*=506**: *Draughting elements*; чертежные элементы.

***N*=507:** *Application interpreted construct: Geometrically bounded surface*; геометрически ограниченные поверхности.

***N*=510**: *Geometrically bounded wireframe*; геометрически ограниченная модель поверхности.

***N*=511**: *Topologically bounded surface*; с топологически ограниченная модель поверхности.

***N*=512**: *Faceted boundary representation*; полигональное представление поверхностей деталей..

***N*=515**: *Constructive solid geometry*; конструктивная геометрия.

Стандарты управления качеством продукции

Стандарты серии ISO 9000 управления качеством промышленной продукции делятся на первичные, вторичные и поддерживающие.

В свою очередь, *первичные* стандарты делятся на внешние и внутренние. Внешние стандарты инвариантны к приложениям, они описывают требования, соблюдение которых гарантирует качество при выполнении контрактов с внешними заказчиками. Внутренние стандарты предназначены для внутреннего использования, они описывают мероприятия по управлению качеством внутри компании.

ISO предлагает следующие внешние стандарты:

ISO 9001 — модель качества, достигаемого при проектировании, производстве, обслуживании;

ISO 9002 — сокращенная по сравнению с ISO 9001 модель (без процессов проектирования);

ISO 9003 — модель качества при финальном тестировании продукции.

Во внешних стандартах содержатся 20 основных требований к качеству, называемых *системными элементами*. Системные элементы разделены на группы, относящиеся к производству, транспортировке и постпроизводственным операциям, документации, маркетингу. Например, при производстве контролируются планирование, процедуры, программы и инструкции для управления и улучшения производственных процессов. При маркетинге контролируются такие системные элементы, как функциональное описание продукции, организация обратной связи с заказчиками (отслеживание и анализ рекламаций).

Вторичные стандарты включают в себя:

ISO 9000 — основные понятия, руководство по применению ISO 9001;

ISO 9004 — элементы систем управления качеством.

Поддерживающие стандарты предназначены для развития и установки систем качества:

ISO 10011 — аудит, критерии для аудита систем качества;

ISO 10012 — требования для измерительного оборудования;

ISO 10013 — пособие для развития руководств по управлению качеством.

Стандарты ISO-14000 предназначены для управления и контроля выполнения промышленными предприятиями экологических требований. Составными частями системы управления влиянием на окружающую среду являются следующие элементы.

— Экологическая политика.

— Планирование (аспекты экологии, требования законов, цели, программа управления).

— Внедрение и функционирование (структура и распределение обязанностей; обучение, понимание и компетентность; коммуникации; документация; контроль операций; учет аварийных ситуаций).

— Контроль и корректирующие действия (наблюдение и измерения; предупреждающие действия; фиксация ситуаций; аудит системы).

— Анализ работы системы.

После того, как принято решение о внедрении стандарта ISO 14001, разрабатывают план внедрения, включающий расписание, требуемые затраты, определяются задействованный персонал, распределение ответственности и ресурсов, внешняя поддержка. Далее формулируется и документально фиксируется политика организации, в частности, предупреждение загрязнений, соответствие законодательству и т.п. Выполняется анализ существующего состояния вопроса на предприятии. На его основе разрабатывается стратегический план внедрения стандарта. В соответствии с планом производственные процессы, продукция и службы предприятия оцениваются с точки зрения взаимодействия с окружающей средой. Рассматриваются такие факторы, как шум, излучения, отходы производства, энергетические воздействия и среди них выделяются актуальные для предприятия контролируемые факторы. Принимаются во внимание законодательство и интересы соседей.

Систему ISO 14001 целесообразно создавать не автономно, а интегрируя ее с другими системами управления предприятием, что обеспечит и облегчит ее сопровождение и постоянное использование.

Список литературы

1. **Липаев В.В., Филинов Е.Н.** Мобильность программ и данных в открытых информационных системах. — М.: Научная книга, 1997.

2. **Норенков И.П., Трудоношин В.А.** Телекоммуникационные технологии и сети. — М.: Изд-во МГТУ им.Н.Э. Баумана, 1998.

3. **Острейковский В.А.** Теория систем. — М.: Высш. шк., 1997.

4. Системы автоматизированного проектирования: Учеб. пособие для втузов: В 9 кн./ Под ред. **И.П.Норенкова**. — М.: Высш. шк., 1986.

5. **Фоли Дж., вэн Дэм А**. Основы интерактивной машинной графики: Пер. с англ. В 2-х кн. — М.: Мир, 1985.

6. **Черненький В.М.** Имитационное моделирование. — М.: Высш. шк., 1990.